(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
15 November 2001 (15.11.2001)

PCT

(10) International Publication Number
WO 01/86444 A1

(51) International Patent Classification[7]: G06F 11/00

(21) International Application Number: PCT/US01/14618

(22) International Filing Date: 7 May 2001 (07.05.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/202,299	5 May 2000 (05.05.2000)	US
60/202,296	5 May 2000 (05.05.2000)	US
60/202,298	5 May 2000 (05.05.2000)	US

(71) Applicant *(for all designated States except US)*: APRISMA MANAGEMENT TECHNOLOGIES, INC. [US/US]; 121 Technology Drive, Durham, NH 03824 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **GREENWALD, Joseph** [US/US]; 13 Cherry Lane, Madbury, NH 03820 (US). **BALL, Scott** [US/US]; 35 Johnson Drive, Newmarket, NH 03857 (US). **BUIA, Christopher, A.** [—/US]; 10 Dean Drive, Dover, NH 03820 (US). **CARON, Jonathan, P.** [—/US]; 59 Garland Road, Nottingham, NH 03290 (US). **TAYLOR, David, K.** [—/US]; 9 Willey Road, Durham, NH 03824 (US).

(74) Agent: **ZITKOVSKY, Ivan, D.**; 6 Freeman Circle, Lexington, MA 02421-7713 (US).

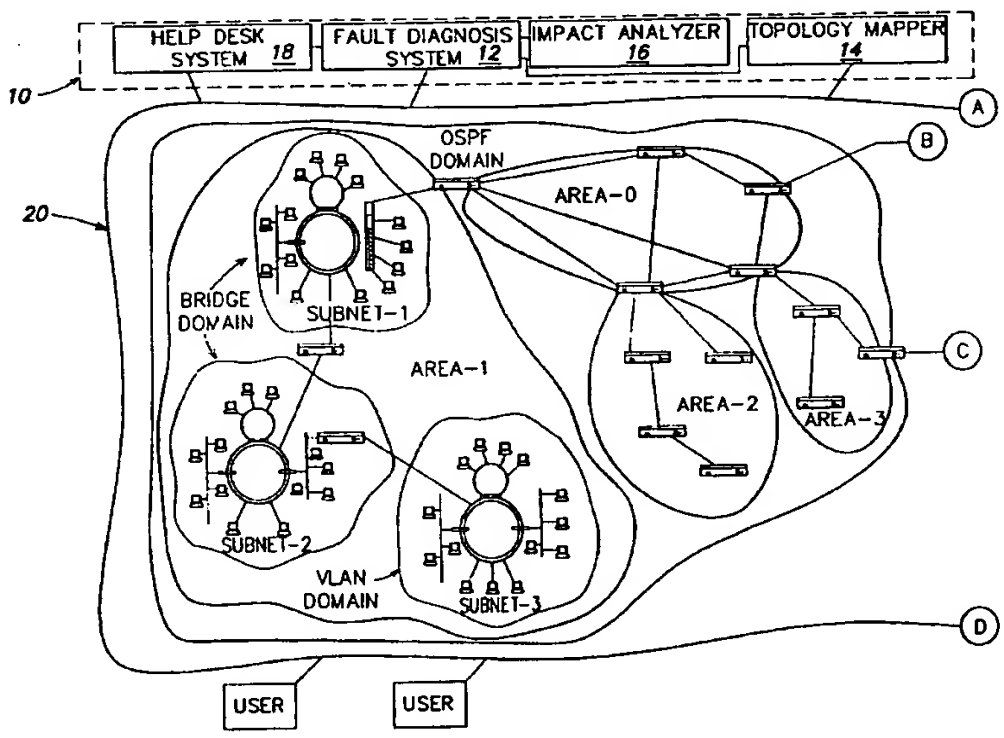
(81) Designated States *(national)*: AU, CA, US.

(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEMS AND METHODS FOR DIAGNOSING FAULTS IN COMPUTER NETWORKS



(57) Abstract: The network management system (10) includes a fault diagnosis system (12) connectable to a communications network (20). The fault diagnosis system includes a fault object factory, a path determination module, and a fault diagnosis engine. The fault object factory is constructed and arranged to receive fault data and create fault objects. The path determination module is constructed to determine, in a communications network, a path related to the fault data. The fault diagnosis engine is constructed to perform, using the determined path, a fault analysis on the fault data to diagnose a fault related to the communications network. The network management may also include a help desk system (18), a topology mapper (14) or an impact analyzer (16).

WO 01/86444 A1

SYSTEMS AND METHODS FOR DIAGNOSING FAULTS IN COMPUTER NETWORKS

This application claims priority from a U.S. Provisional Application 60/202,296, entitled ``Construction of a Very Rich, Multi-layer Topological Model of a Computer Network for Purposes of Fault Diagnosis,'' filed on May 5, 2000, and claims priority from U.S. Provisional Application 60/202,299, entitled ``A method for diagnosing faults in large multilayered environments guided by path and dependency analysis of the modeled system,'' filed on May 5, 2000, and claims priority from U.S. Provisional Application 60/202,298, filed on May 5, 2000, entitled ``Method and apparatus for performing integrated computer network, system, and application fault management,'' all of which are incorporated by reference in their entireties.

General Description

The invention relates to detecting, diagnosing and managing faults in communications networks.

The construction of computer networks started on a large scale in the 1970's. Computer networks link personal computers, workstations, servers, storage devices, printers and other devices. Historically, wide area computer networks (WANs) have enabled communications across large geographic areas, and local area networks (LANs) communications at individual locations. Both WANs and LANs have enabled sharing of network applications such as electronic mail, file transfer, host access and shared databases. Furthermore, WANs and LANs have enabled efficient transfer of information, and sharing of resources, which in turn increased user productivity. Clearly, communications networks have become vitally important for businesses and individuals.

Communications networks usually transmit digital data in frames or packets created according to predefined

protocols that define their format. Data frames include headers (located at the beginning and containing addresses), footers (located at the end of the frames), and data fields that include the transmitted data bits (payload). Data frames may have a fixed or variable length according to the used protocol or network type.

A communications network transmits data from one end station (i.e., a computer, workstation, server etc.) to another using a hierarchy of protocol layers (i.e., layers that are hierarchically stacked). In the communication process, each layer in the source communicates with the corresponding layer in the destination in accordance with a protocol defining the rules of communication. This is actually achieved by transferring information down from one layer to another across the layer stack, transmitting across a communication medium, and then transferring information back up the successive protocol layers on the other end. To facilitate better understanding, however, one can visualize a protocol layer communicating with its counterparts at the same layer level.

The open system interconnection (OSI) model has seven layers that define the rules for transferring information between the stations. A physical layer (Layer 1) is responsible for the transmission of bit streams across a particular physical transmission medium. This layer involves a connection between two endpoints allowing electrical signals to be exchanged between them.

A data link layer (Layer 2) is responsible for moving information across a particular link by packaging raw bits into logically structured packets or frames. Layer 2 ensures good transmission and correct delivery by checking errors, re-transmitting as necessary, and attaching appropriate addresses to the data sent across a physical medium. If a destination computer does not send an acknowledgment of frame receipt, Layer 2 resends the frame. The contention access methods (e.g., CSMA/CD, and Token

Passing) are regarded as Layer 2 activities. Layer 2 may be further divided into two sub-layers: Logical Link Control (LLC) and Media Access Control (MAC). The MAC sublayer defines procedures the stations must follow to share the link and controls access to the transmission link in an orderly manner. The MAC sublayer defines a hardware or data link address called a MAC address. The MAC address is unique for each station so that multiple stations can share the same medium and still uniquely identify each other. The LLC sublayer manages communications between devices over a single link of the communications network.

A network layer (Layer 3) is set up to route data from one network user to another. Layer 3 is responsible for establishing, maintaining, and terminating the network connection between two users and for transferring data along that connection. Layer 3 addresses, messages, and determines the route along the network from the source to the destination computer. Layer 3 manages traffic, such as switching, routing, and controlling the congestion of data transmissions.

A transport layer (Layer 4) is responsible for providing data transfer between two users at an agreed level of quality. When a connection is established, this layer is responsible for selecting a particular quality of service (QoS), for monitoring transmissions to ensure the selected QoS, and for notifying the users if the QoS deteriorates. Layer 4 also provides for error recognition and recovery, repackaging of long messages into smaller frames of information, and acknowledgments of receipt.

A session layer (Layer 5) focuses on providing services used to organize communication and synchronize the dialog that takes place between users and to manage the data exchange. The primary concern of Layer 5 is controlling when users can send and receive concurrently or alternately.

A presentation layer (Layer 6) is responsible for the presentation of information in a way that is meaningful to

network users. This may include character code transmission, data conversion, or data compression and expansion.

Layer 6 translates data from both Layer 5 and from Layer 7 into an intermediate format and provides data encryption and compression services. Layer 7 is an application layer that provides means for application processes to access the system interconnection facilities in order to exchange information. This includes services used to establish and terminate the connections between users and to monitor and manage the systems being interconnected, as well as the various resources they employ.

As data is passed down through the layers, each layer may or may not add protocol information to the data, for example, by encapsulating frames with a header or removing the header, depending on the direction in the protocol stack. The individual protocols define the format of the headers.

MAC address includes a source address and a destination address, which have a predefined relationship to a network station. Higher network layers provide a network address that has a logical relationship established by a network administrator according to a predetermined network addressing arrangement. The assigned network address conveys information that can be used by a router when routing frames through the internetwork. If the network address is hierarchical, a router may use a portion of the address to route the packet to a higher-level partition or domain in the internetwork. Some protocols are hierarchical others are not so hierarchical routing may or may not be available.

The global network may be subdivided into IP networks, which in turn may be subdivided into subnets. An IP address includes a network number (assigned by IANA), a subnet number (assigned by a network administrator), and a host that identifies an end station. The host number may be assigned by a network administrator, or may be assigned

dynamically. This is a form of hierarchical addressing that is used by IP routing algorithms to perform hierarchical or prefix routing operations. Routing algorithms maintain information of all higher-level routing environments in routing tables for domains by recording their shortest unique address prefixes.

A station may support more than one network layer protocol. Such station has multiple network addresses and multiple protocol stacks that present the same MAC address on a port for the different protocols. Thus, a multi-protocol stack station connected to both an IP and an IPX network includes an IP network address and an IPX network address.

A communications network may include a number of network entities (or nodes), a number of interconnecting links and communication devices. A network node is, for example, a personal computer, a network printer, file server or the like. An interconnecting link is, for example, an Ethernet, Token-Ring or other type network link. Communication devices include routers, switches, bridges or their equivalents. As computer networks have grown in size, network management systems that facilitate the management of network entities, communication links and communication devices have become necessary tools for a network administrator.

A bridge or a switch is a Layer 2 entity that is typically a computer with a plurality of ports for establishing connections to other entities. The bridging function includes receiving data from a port and transferring that data to other ports for receipt by other entities. A bridge moves data frames from one port to another using the end-station MAC address information contained in the switched frames. Switches interconnect the communication media to form small domains of stations, such as a subnetwork. Subnetworks or subnets provide an organizational overlay to an internetwork that facilitates

transmission of data between the end stations, particularly for broadcast transmissions. The subnet functions to limit the proliferation of broadcast frames to stations within a broadcast domain.

A router is an intermediate station that interconnects domains or subnets by providing path from a node on a first network to a node on a second network. There are single protocol or multi-protocol routers, central or peripheral routers, and LAN or WAN routers. A peripheral router connects a network to a larger internetwork, and thus may be limited to a single protocol. A central router may be connected to a different board in a server or a hub and thus usually has a multi-protocol capability.

A router provides the path by first determining a route and then providing an initial connection for the path. A router executes network routing software that depends on the used protocol. A router can work with different data-link layer protocols and thus can connect networks using different architectures, for example, Ethernet to Token Ring to FDDI. Furthermore, there are routers of several levels, wherein, for example, a subnetwork router can communicate with a network router. Organizing a communications network into levels simplifies the routing tasks since a router needs to find only the level it must deal with. The use of different network levels is shown in Fig. 1.

In general, a global communications network connects devices separated by hundreds of kilometers. A LAN covers a limited area of maximum several kilometers in radius connecting devices in the same building or in a group of buildings. LANs usually include bridges or switches connecting several end-stations and a server. In a LAN, a bridge or a switch broadcasts traffic to all stations. Until a few years ago, a LAN was user-owned (did not run over leased lines) with gateways to public or other private networks. When a user moved or changed to an end-station at another location on the network, a network administrator had

to rewire and reconfigure the user's station. This has changed with the introduction of virtual LANs.

A virtual LAN (VLAN) is a logical Layer 2 broadcast domain, which enables a logical segmentation of the network without changing the physical connections. A VLAN enabled switch segments the connected stations into logically defined groups. Broadcast traffic from a server or an end-stations in a particular VLAN is replicated only on those ports connected to end-stations belonging to that VLAN. The broadcast traffic is blocked from ports with no end-points belonging to that VLAN, creating a similar type of broadcast containment that routers provide. VLANs may also be defined between different domains connected by a router. In this case, the router passes network traffic from one domain to the other (as done without defining a VLAN), and passes network traffic from one VLAN to the other. The router also passes network traffic between VLANs that are in the same domain because VLANs do not normally share user information. The router is configured as a member of all VLANs.

The VLAN designation is assigned (e.g., programmed) to each internal port and is stored in a memory portion of the switch. Every time a message is received by a switch on an internal port, the VLAN designation of that port is associated with the message.

Virtual Private Networks (VPNs) have been designed to interconnect end-stations that are geographically dispersed. For example, owners of large communications networks can provide centralized management services to small and medium sized businesses. The provider can configure VPNs that interconnect various customer sites in geographically separate locations. These VPNs offer privacy and cost efficiency through sharing of network infrastructure. Various VPNs have been proposed with various degrees of security, privacy, scalability, ease of deployment and manageability.

A global communications network may use a different

levels different routing and connection management protocols such as International Standards Organization (ISO) Open Systems Interface (OSI) Intermediate Systems to Intermediate Systems (IS-IS), and Internet Open Shortest Path First (OSPF) protocols are used for connectionless routing of data frames. Asynchronous Transfer Mode (ATM) Forum Private Network-Network-Interface (PNNI) protocol is used for connection oriented multi-media services. The routing protocols identify a network node using a global address of a Route Server Element (RSE). The RSEs generate routing that identify optimal routes for communication throughout the network. The RSE is responsible for administration of the algorithms that enable a node to keep its view of the network topology and performance metric current, referred to as Routing Information Exchange (RIE). Thus an RSE usually acts as a central element for the routing of traffic through the node.

In general, the use of WANs, LANs, VPNs, and VLANs has increased the number and complexity of communications networks. These networks continuously evolve and change due to growth and introduction of new interconnections, topologies, protocols, or applications. Furthermore, most networks have redundant communication paths to prevent portions of the network from being isolated due to link failures. Also, multiple paths can be used simultaneously to load-balance data between the paths. However, redundant paths can also introduce problems such as formation of loops. Furthermore, network performance can degrade due to improper network configurations, inefficient or incorrect routing, redundant network traffic or other problems. Network hardware and software systems may also contain design flaws that affect network performance or limit access by users to certain of the resources on the network. These factors make network management complex and difficult.

A network management process controls and optimizes the efficiency and productivity of a communications network. A

network management station manages the network entities (e.g., routers bridges switches, servers, storage devices, computers, printers) using a network management protocol such as a Simple Network Management Protocol (SNMP), Internet Control Message Protocol (ICMP), or another network management protocol known in the art. Using a network management protocol, the network management station can deliver information or receive information by actively polling the network entities or by receiving unsolicited information from the network entities. Using SNMP, a network management station can executes a set, get, or get-next functions to set and retrieve information from a network entity. This information may be stored within the polled network entity as Management Information Base (MIB). The network management station can receive unsolicited information from a network entity in the form of an SNMP trap. Network entities may send SNMP traps to the network management station when a problem in the network or network entity occurs.

A network management station may be implemented using any general purpose computer system, which is programmable using a high-level computer programming language or using specially programmed, special purpose hardware. The hardware includes a processor executing an operating system providing a platform for computer programs that run scheduling, debugging, input-output control, accounting compilation, storage assignment, data management, memory management, and communication control and other services. The application programs are written in high level programming languages.

A network management station can include a network manager unit, a network communication interface, a data acquisition unit, a data correlation unit, and a graphical user interface. The data correlation unit interprets data received through the data acquisition unit and presents the interpreted data to a user on the graphical user interface. The network communication interface may include transport

protocols and LAN drivers used to communicate information to the communications network. The transport protocols may be IPX, TCP/IP or other well-known transport protocols. The LAN drivers may include software required to transmit data on a communications network through the network interface. The LAN drivers are generally provided by the manufacturer of the network interface for a general purpose computer for the purpose of communicating through the network interface. The network manager unit may be an SNMP network manager/agent implementing SNMP functions, or another type of network manager unit performing associated management functions. The network manager unit utilizes the network communication interface to transfer requests to network entities over a communications network.

A network management station may use a network management agent residing on a network entity. The network management agent may be a software process running on a processor or may be special purpose hardware. The network management agent may be an SNMP agent (or ICMP agent?), which may include a data collection unit, a network manager unit, and a network communication interface for communication as described above. For example, this communication may use network management functions such as SNMP functions. Alternatively, a network management agent, residing on a network entity, may include a data correlation unit, a data collection unit, a network manager unit and a network communication interface for communication.

In the network management station, the data correlation unit interprets data received through data acquisition unit. The data correlation unit may include an interface processor, a state processor, a hierarchical database, and one or more sets of object rules. The interface processor communicates with graphic libraries residing in the graphical user interface to present interpreted data to a user. The interface processor performs graphic functions associated with objects related to a network map or model.

The graphic functions generate visual notifications to a user. The state processor performs correlation functions of the data correlation unit by requesting and receiving network management information from the data acquisition unit. The data acquisition unit stores data and requests information from the network manager.

In the network agent, the data correlation unit interprets data received by data collection unit. The state processor performs correlation functions of the data correlation unit by requesting and receiving network management information from the data collection unit. The data collection unit stores data and requests information from the network manager. In the network agent, the data collection unit can collect data from the network entity directly through its own network manager. The data collection and acquisition units can invoke network management commands within network manager libraries, which send and retrieve information from a network entity. These commands may be SNMP functions mentioned above, or ICMP functions. The state processor interprets data received from the network manager libraries in accordance with object rules. The object rules define how the state processor combines or parameterizes different properties of a network entity. The state processor can produce parameterized properties from the object rules.

Communications networks are becoming increasing complex, both physically and logically. It is no longer adequate for network management stations to merely report problems of unreachability. Network and application service providers need to maintain availability, while also ensuring optimal performance. Therefore, there is a need for providers to detect and analyze soft failures (like performance brownouts) and failures before they affect service availability. Furthermore, there is a need to detect and diagnose loss of redundancy, denial of service attacks, inadvertent misconfigurations, and degradation of

services.

Summary of the Invention

The present invention is directed to a system, a method and a product (which can be stored in a computer-readable storage medium) for detecting, diagnosing or managing faults in communications networks. In our description we define a fault as a complete or partial failure, service disruption or deterioration of service quality. For example, network fault conditions includes no-access faults, slow access (bottlenecks), redundancy loss, network blasters (i.e., entities emitting excessive amounts of traffic), obstacles (e.g., when one entity prevents another entity from achieving maximum throughput), stale or misplaced static routing information, incorrect network policies, packet filtering or ACL misplacement, or denial of service attacks.

According to one aspect, a method and system for diagnosing a fault in a communications network including a fault object factory, a path determination module, and a fault diagnosis engine. The fault object factory is constructed and arranged to receive fault data and create fault objects. The path determination module is constructed to determine, in a communications network, a path related to the fault data. The fault diagnosis engine is constructed to perform, using the determined path, a fault analysis on the fault data to diagnose a fault related to the communications network.

Preferred embodiments of this aspect include one or more of the following features: The system may include a fault detector constructed and arranged to detect a fault in a monitored entity. The system may include a fault repository constructed and arranged for storing and accessing fault objects. The system may include a fault handler constructed to perform analysis of the fault object. The fault handler includes a fault handler tester. The fault handler includes a diagnoser fault handler constructed

and arranged to change a fault state of the fault object. The fault repository includes a fault object and a fault association object. The fault object includes a description, a processing state, and test result object. The test result object includes a description, a target entity, test data, and a result state.

According to another aspect, a network management system includes a fault diagnosis system connectable to a communications network. The fault diagnosis system includes a fault object factory, a path determination module, and a fault diagnosis engine. The fault object factory is constructed and arranged to receive fault data and create fault objects. The path determination module is constructed to determine, in a communications network, a path related to the fault data. The fault diagnosis engine is constructed to perform, using the determined path, a fault analysis on the fault data to diagnose a fault related to the communications network.

The network management may also include a help desk system, a topology mapper or an impact analyzer.

According to yet another aspect, a method and system for diagnosing a fault in a communications network including means for receiving fault data; means for determining path in the communications network related to the fault data; and means for performing a fault analysis, using the determined path, on the fault data to diagnose a fault related to the communications network.

Preferred embodiments of this aspect include one or more of the following features: Means for performing the fault analysis may include means for performing a root cause analysis. Means for determining path includes means for performing a domain-specific algorithm. The path determination includes dynamically generating a trace route by employing a path-tracing algorithm. Means for determining path includes means for employing data stored in a topology mapper. Means for determining path includes

using means for routing and switching algorithms used by the network for data forwarding. Means for determining path includes means for performing a layer 3 path determination, which may include finding a router in a subnet and using the router as a starting point for path generation.

Means for receiving fault data may include creating a fault object in a fault object factory using the fault data. The system may use a detector (e.g., an agent) for registering fault data and providing the fault data for analysis. Means for performing root cause analysis include means for triggering a specific fault handler such as a diagnoser fault handler or a tester fault handler. The diagnoser fault handler may be designed to handle QoS problems. The diagnoser fault handler may b designed to handle problems in virtual private networks. The diagnoser fault handler may be designed to handle problems in multi-cast groups. The diagnoser fault handler may b designed to handle wireless connectivity problems. The diagnoser fault handler may be designed to handle problems related to cable access or DSL access. The diagnoser fault handler may be designed to ``know'' about components of a particular network layer and rely on diagnosis capabilities of their constituent subcomponents to examine deeper through the layers.

According to yet another aspect, when modeling the communications network, the system generates source to destination paths by following the standards-based routing and forwarding algorithms employed by the network devices. Then, by examining the actual application dependencies and the network elements in the path, the system efficiently diagnoses various problems thereby minimizing the burden on an already degraded network environment. Additionally, the combination of application dependencies and path analysis allows the system to generate the impact of network degradation. Based on the accessibility of

these applications, users can be informed about important problems. Furthermore, the network management system can intelligently adjust its monitoring of the degraded section of the network until the problem is remedied. Finally, the system can use path analysis as a tool to verify that problems have been fixed.

The present system provides a collection of autonomous or integrated network fault detectors and service disruption detectors that report symptoms of network faults. Based on the reported symptoms, a fault object factory creates fault objects in a shared fault repository. A fault diagnosis engine (which is a generic fault-processing engine) triggers fault handlers that include fault testers and fault diagnosers. The fault diagnosers are registered for notification of faults of a specific type entering a specific state. Fault diagnosers use the rich topology model of the network and services, efficient data path determination, and innovative analysis techniques to find related faults. By finding and analyzing related faults, the system ultimately finds the root cause fault. The fault diagnosers may also detect new related or unrelated faults and the fault object factory then creates new fault objects for the fault repository. Using the fault objects the system constructs a fault causality tree. As a result, the root cause fault is diagnosed and the impact of the fault is determined.

Brief Description of the Drawings

Fig. 1 shows diagrammatically several network management modules connectable to a communications network.

Figs. 2 and 2A are block diagrams of a fault

management and diagnosis process.

Fig. 3 is a block diagram of modules employed in a fault management and diagnosis system.

Figs. 3A and 3C are block diagrams of objects employed in the fault management and diagnosis system of Fig. 3.

Fig. 3B is a block diagram of a fault repository module employed in the fault management and diagnosis system of Fig. 3.

Fig. 4 is a flow diagram that illustrates a triggering mechanism for fault handlers by a fault diagnosis engine shown in Fig. 3.

Figs. 5 and 5A are a block diagrams depicting processing states of a fault during fault diagnosis performed by a fault management and diagnosis system.

Fig. 6 illustrates a portion of the communications network shown in Fig. 1.

Figs. 6A, 6B, 6C, 6D, 6E and 6F are flow diagrams used for path dependent fault diagnosis.

Figs. 7 and 7A are block diagrams of faults diagnosed in the portion of a communications network shown in Fig. 6.

Description of Preferred Embodiments

Fig. 1 shows diagrammatically a network management system 10 including a fault diagnosis system 12, a topology mapper 14, an impact analyzer 16 and a help desk system 18. The network management system communicates with a communications network 20 (or application service). The network includes a set of interconnected network elements such as routers, bridges, switches, and repeaters. These network elements provide transportation of data between end stations. Furthermore, there are computers known as servers that provide services such as e-mail, accounting software, sales tools, etc. Typically, data is transmitted electronically or optically, and network elements can forward data in packets, frames or cells to the intended

destination. Servers include network adapters and/or software that interpret the electronic or optical data packet into the data elements and pass these elements to the appropriate application being hosted.

The network management system 10 includes a commercially available processor (for example, Pentium microprocessor manufactured by Intel Corporation) executing an operating system providing an operating environment for a network management program. The processor and the operating system provide a computer platform for which application programs are written in higher level programming languages. The computer (or application host) interfaces with permanent data storage, such as a magnetic or optical disk drive, a disk array, non-volatile RAM disk, or a storage area network, which maintain data files such as user configurations and policies. In general, the network management program may be configured as a generic software application residing in any commercially available computing platform.

Preferably, fault diagnosis system 12, topology mapper 14, and help desk system 18 are software applications written in Java and running on any computer with a Java Runtime Environment (JRE). For example, a Dell laptop computer with an Intel Pentium processor running the Windows 2000 operating system, or a Sun Ultra 60 computer running Solaris v. 2.7. Alternately, fault diagnosis system 12, topology mapper 14, and help desk system 18 are developed in any object oriented or structured programming language, and compiled for execution on any one or many computer platforms, or could be implemented on a neural network computing device.

The computer has a network adaptor that provides communication (preferably, but not necessarily, IP) to the users on the network. The fault diagnosis engine application may share a host with help desk system, and/or the topology mapper, or each can run on a separate host, in

which case they communicate using a network adaptor. Topology mapper 14 determines the network topology and creates a model. The permanent data storage holds data files that describe the current network topology, and configuration files that control the performance of topology mapper 14. A user is an end station, interfaced to access the network or services, used by a person who is using the network, or is using services provided by the network.

The network management system 10 performs a fault management process 30 shown in Fig. 2. The entire process is part of a phased, componentized, but interconnected method, wherein all aspects of fault management are performed. The fault management process of Fig. 2 includes the following seven phases: fault detection 32, diagnosis 40, impact analysis 50, prioritization 60 presentation 70, recourse 80, and resolution 90.

Fault detection process 32 (performed by fault detectors 130 shown in Fig. 3) is the most basic part of the fault management system. Fault detectors 130 detect raw fault data. Fault detectors 130 receive information by SNMP polling, SNMP trap handling, performance monitoring, historical trend analysis, device configuration monitoring, application and system-level management tools, and help desk trouble tickets. Fault detection process 32 can also add information to the raw fault data enabling improved diagnosis of the fault. The fault data are assembled into fault objects.

Fault diagnosis 40 occurs after a "detected" fault is entered into the system. A fault detection and management system 100 processes and correlates detected faults with other faults to determine their relationship. Fault detection system 100 finds one or more "root cause" faults and isolates these faults. Furthermore, the system can optionally suppress other symptomatic faults that were "caused" by the root cause faults. Fault diagnosis 40 can be performed in a single step or can involve many techniques

such as examining device neighbor knowledge, tracing the route of management data, examining route tables and ACLs, etc.

Fault impact analysis 50 determines the "scope" of the analyzed fault. After receiving a root cause fault determined, by fault diagnosis 40, impact analysis 50 determines the consequences of this fault. This determination includes analyzing the network services affected by the fault, the users affected by the fault, and any other ramifications the fault has on network 20, or the application being managed. Furthermore, impact analysis 50 may involve analyzing various logical layers that exist in a communication network and correlating a fault with its possible consequences at each layer. Impact analysis 50 may use a fault causality tree located in a fault repository 140 (Fig. 3). The interpretation schemes include analyzing how a network fault affects services like web servers or e-mail, examining how a misconfigured router running OSPF affects the users in each area, etc.

The network management system may also perform fault prioritization 60. After a fault has been diagnosed and its impact analyzed, the fault may be prioritized. Fault prioritization 60 assigns a priority/severity to each fault object and this is used to determine the proper presentation of the fault to a user. Fault prioritization process 60 may include multiple methods based on the type and scope of the fault such as examination of service level agreements and how the fault violates them, mission critical device analysis, and fault scope.

The network management system may also perform fault presentation 70. Fault presentation 70 provides the mechanism by which the system alerts a user that a fault has occurred. Fault presentation process 70 presents all information about the fault in a user-friendly manner. Fault presentation 70 may include steps and processes the systems used to diagnose the fault, thus allowing a user to

verify the diagnosis and "trust" the system to accurately diagnose faults. Fault presentation 70 may also include a network monitoring alarm system.

The network management system may also include fault recourse 80. Fault recourse 80 provides a way in which a user can change the network management based on a given fault. For example, fault recourse 80 may involve reducing or stopping polling of devices downstream from a fault, reconfiguring connectivity modeling, script invocation to fix a misconfigured static route, or configuring user groups for a different email server.

The network management system may also include fault resolution 90. After presenting a fault to a user and fixing the problem, problem resolution 90 records the process for future fault detection and diagnosis. Fault resolution 90 can automatically trigger for any single resolved fault a re-evaluation of associated faults in the system. This re-evaluation proactively assesses the full scope of a resolved fault. If an associated fault is still not resolved, diagnosis can be re-started to determine the cause. This process is facilitated by the use of the fault causality tree created as a result of fault diagnosis process 40.

Fig. 2A shows diagrammatically in detail fault diagnosis process 40. A detected fault enters the fault detection and management system and a fault object is created (step 42). The fault diagnosis engine triggers appropriate fault handlers (step 43). A diagnoser fault handler performs path determination algorithms and domain specific algorithms as described in connection with Fig. 6A through 6F. Then, the diagnoser fault handler generates possible faults that may be causes of the entered fault (step 44). For each generated, possible fault, the fault diagnosis engine triggers appropriate tester fault handlers (step 45). Each tester fault handler performs vendor-specific and domain-specific

tests to determine the existence of one or several possible faults. Next, the tester fault handler records test results (step 46). If possible additional faults exist, the fault diagnosis engine continues to trigger tester fault handlers and diagnoser fault handlers (step 49). If there are no other possible faults, the fault diagnosis engine has isolated the fault and the system proceeds to impact analysis 50.

Fig. 3 shows fault diagnosis system 100, which includes five main parts: a fault diagnosis engine 101, a fault object factory 110, fault detectors 130, a fault repository 140, and handlers 150. Fault detection and management system 100 (or fault diagnosis system) has the ability to receive detected fault information from multiple sources, control the management of the faults, and produce a root cause analysis. Furthermore, the system also provides a mechanism for performing fault correlation and impact analysis. The impact assessment is not limited to the impact of the communications network, but may include disruptions in services or applications that depend on the network infrastructure.

Fault object factory 110 receives data from fault detectors 130 and creates fault objects 112 shown in Fig. 3A. Each fault object 112 is associated with a fault type and there may be many fault types. Furthermore, each instance is a separate occurrence of a problem, potential problem, or condition of a communication network or an element located in the communication network (such as a misconfiguration, a degradation of service, physical failure or other).

Referring to Fig. 3A, the entire architecture of the fault detection and management system based on fault objects 112, which are records representing a detected problem, a potential problem, or a condition. Fault object 112 includes information about a detected fault, that is, includes a description of the problem or condition stored in

field 114, time and date of the reported problem 116, a fault processing state 118, and one or more test result objects 120. The fault structure includes a context that is a mechanism for sharing varying amounts of data related to the fault; these amounts may exist between each instantiation of a type of fault.

Referring to Fig. 3, fault detector 130 detects a problem or potential problem on an entity in a managed system. Fault detector 130 provides a record of the condition to fault object factory 110, which generates fault object 112. Fault detector 130 can monitor an entity or receive unsolicited notification from an entity when a problem occurs, according to different methods known in the art. Fault detector 130 may perform a test and may provide to fault object factory 110 data with the results of the performed tests. Fault detector 130 may share a host with fault diagnosis engine 101, or may reside externally as an agent.

Referring to Fig. 3B, fault repository 140 is the component used by a fault detection and management system 100 to store and access fault information. fault repository 140 stores every fault object 112 present in the system. Each component of the system (detection, diagnosis, etc.) can enter new fault objects into fault repository 140 and access any fault object 112. Preferably, fault repository 140 includes a table structure with services capable of searching and locating existing faults.

Fault repository 140 also includes fault associations 142, which provides a mechanism for relating faults to one another. Specifically, each defined fault association relates two fault objects. One fault object is on the left side of the association, and the other fault object is on the right side as shown for fault trees below. The semantics of an association are defined by the type of the association. New fault association types can be defined and added to the system, preferably using Interface Description

Language (IDL) definitions of an interface for a service that uses the Common Object Request Broker Architecture (CORBA) transport protocol.

Referring again to Fig. 3, each fault handler 150 performs a designated type of work as a result of a fault object entering a certain processing state (shown in Fig. 5). Fault handlers 150 may exist internal to the system, or reside externally in a separate process. Fault handlers 150 are registered for a particular fault type and state and, as part of the registration process, each fault handler 150 has an integer priority value. Then, fault handlers 150 are sorted by their priority values so that a fault handler with the lowest priority value is triggered first and subsequent handlers are triggered in sequence, as described below. One type of fault handler 150 can test a fault object and create a test result record. Furthermore, fault handler 150 may create additional types of fault objects, create associations between fault objects, correlate fault objects that indicate a similar problem, or perform impact analysis on a fault object to determine the scope of a problem. A tester fault handler 152 performs a selected test on a fault. A diagnoser fault handler 154 receives notification when a particular type of a fault enters a state. Diagnoser fault handler 154 may change the fault state (Fig. 5) and may associate the fault with other potential faults.

Fault diagnosis engine 101 is the central component of fault detection and management system 100 since it drives the management and diagnosis of faults. Fault diagnosis engine 101 provides a generic mechanism for fault handlers 150 to register for changes in the processing state of faults of a given fault type. Fault diagnosis engine 101 may employ any mechanism to specify registrations. The preferred implementation of fault diagnosis engine 101 uses XML (Extensible Markup Language) technology.

Referring to Fig. 4, when a fault transitions to a state for which a handler has registered, the engine

triggers the handler to perform its work. Fault diagnosis engine 101 can trigger one of fault handlers 150 arbitrarily or may use some triggering mechanism. Preferably, fault diagnosis engine 101 uses a priority mechanism to order the triggering of fault handlers that are sorted by their priority value (by triggering first a fault handler with the lowest value).

Fault detection and management system 100 uses fault processing states for analyzing faults. A fault's processing state represents its status in the fault management process and provides a mechanism to control the management of the fault. A fault can have a large number of possible states, and a fault can transition from state to state using different ways, as shown in Figs. 5 and 5A. Preferably, the system utilizes a fault type hierarchy in which generic base fault types are defined and from which, new more specific fault types can be derived. Each fault, which exists in the system, is of some pre-defined fault type.

Referring to Fig. 3C, a test result object 120 includes a record of test results that were performed to determine the existence of the problem or condition for which the fault was created. Test result object 120 includes a textual description of the test (field 122), data obtained from the target of the fault (field 123), test data (field 124), any thresholds and parameters used in determining the test result (field 125). Test result record 125 also contains a state representing the status of the test.

While performing its work on a fault object, a fault handler may cause the processing state of the fault to be changed. In this case, no other handlers for the current state are triggered. Fault diagnosis engine 101 obtains the handlers for the new state and resumes triggering with the new handlers when the current handler completes its work.

Fig. 4 illustrates the triggering mechanism using a flow diagram. Fault diagnosis engine 101 provides a

triggering mechanism and allows the fault handlers to control and manage the diagnosis process. The fault handlers are responsible for transitioning faults between processing states, determining causality and performing resolution and/or re-evaluation of faults.

Referring to Fig. 5, fault diagnosis engine 101 utilizes processing states of a fault to control the flow of diagnosis for that fault. As described above, fault handlers 150 are triggered for a fault based on the current processing state. The transition diagram of Fig. 5 defines the following processing states: An initial state 180 begins the life-cycle of a fault object. A detected state 182 indicates that an external fault detector 130 or an internal handler 150 positively determined the condition (that the fault represents) as a problem. A testing state 184 indicates the fault is unverified; that is, a condition that the fault represents requires testing to determine if it is a problem. A completed state 184 indicates that fault diagnosis has completed for the fault.

Fault diagnosis engine 101 may allow fault handlers 150 to directly transition a fault between states, wherein preferably the processing state is hidden from fault handlers 150. The engine transitions a fault's processing state based on the state of the current result of the fault as provided by the handlers. There are the following three test result states (shown in Fig. 5A): PROBLEM indicates a test has identified the fault to be a problem; NO_PROBLEM indicates a test has verified the condition that the fault represents does not or no longer exists; and UNKNOWN indicates a test could not be completed for some reason or the condition that the fault represents requires verification.

Fig. 5A illustrates transition of the processing states (shown in Fig. 5) based on test results of an analyzed fault. For example, fault diagnosis engine 101 triggers tester fault handler 152 (Fig. 3) for testing state 182 and

fault handler diagnoser 154 for detected state 182. Furthermore, fault handler diagnoser 154 may also be triggered for testing state 184 if there are no tester fault handlers that can perform a direct test. There may also be fault handlers for completed state 184, which would not perform diagnosis, but would perform other tasks such as correlating faults that share a common root cause (described below) or notifying a presentation system to display the diagnosis results when performing presentation process 70.

Fault diagnosis engine 101 may employ further rules that govern the triggering of fault handlers when there are multiple handlers (or types of handlers) for a particular processing state. If there are multiple types of handlers, the engine may impose an ordering such that all handlers of one type are triggered before any handlers of another type. Furthermore, if a handler provides a concrete result, as defined by the various result states, the engine may suppress remaining handlers of that type from being triggered and/or may suppress handlers of other types.

According to the preferred embodiment, since there may be both tester fault handlers 152 and diagnoser fault handlers 154 registered for testing state 184, fault diagnosis engine 101 imposes a rule that all tester fault handlers are triggered before any diagnoser fault handler. This is because a tester fault handles can directly determine the existence or nonexistence of a problem, but a diagnoser fault handler cannot. In addition, if a tester fault handlers or diagnoser fault handler provides a concrete result, then fault diagnosis engine 101 suppresses remaining handlers for the current processing state. A concrete result is one whose state is either PROBLEM or NO_PROBLEM. A result state of UNKNOWN is not concrete, that is a result could not be positively determined, as shown in Fig. 5A.

To perform automated network fault analysis, fault detection and management system 100 utilizes different types of network domain fault handlers that are responsible for

handling faults in a particular network domain. The network domain fault handlers include fault testers 152 and fault diagnosers 154 (generically described above). Each network domain fault diagnoser understands the model of its domain and is designed to determine possible causes of a network problem. Preferably, the system uses a specific type of network domain fault diagnoser that performs path determination in the communications network. Then, the determined path elements are combined with the possible problems, tested and analyzed to determine the possible causes of the detected fault.

Topology mapper 14 (or another topology source) provides a network topology including the location of key services (such as e-mail, DNS, web-server, etc.). The system uses the network topology to dynamically generate a thorough trace route using a path-tracing algorithm. The path-tracing algorithm is based on the actual routing and switching algorithms used by the network itself for data forwarding. (The algorithms are published in public standard documents including draft proposals - from IETF, IEEE, ATM Forum, and others.) Therefore the same routing and forwarding decisions that each device makes, in sending packets along the path from the source to the destination, are simulated to create the path used for fault analysis. Since the network is modeled, both physically and logically (Autonomous Systems, OSPF Areas, ATM clouds, VLANs, etc.), network traversal is as efficient as the network itself.

Referring to Fig. 6 using logical layering created by the different routing protocols, a Layer 3 path determination algorithm begins by finding the router that introduced the subnet (specified as the observation point of the fault in the fault object) into the modeled domain. This router and source subnet are the starting point for the path generation. If the destination is in the same subnet as the source, then the subnet model is used to generate a path between the entities - which may result in examining

the spanning tree connectivity between the switches and bridges. If the destination is in another subnet, the system executes a router (layer 3) path analysis. If the source is modeled in the system, the algorithm finds the gateway used by the source workstation.

The internal model of the chosen router is used to simulate the path decision to reach the destination. If the router is running OSPF, then the path determination uses the OSPF's shortest-path-first algorithm. Depending on the routing protocol used, a particular diagnoser is invoked to simulate the behavior of the routing decision to choose the next hop router. Moving from hop to hop (in an internal model) the algorithm generates a complete path from the source to the destination. In a subnet, the algorithm uses the Spanning Tree and Bridged LAN models to generate Layer 2 paths between the routers.

Each logical grouping (for example, OSPF route domain, subnets, ATM clouds, etc.) is analyzed by an associated diagnoser. These logical models can take on specific behavior indicative of the protocol they simulate by way of the associated fault diagnoser.

The automated fault analysis is based on a fault handler designed to handle a specific application or technology and on a model of the network environment. The specific fault handler includes both a diagnoser fault handler and a related tester fault handler. For web-based applications, the system includes a fault diagnoser designed to handle web related faults. For e-mail applications, the system includes a fault diagnoser designed to handle e-mail detected problems.

In the preferred implementation, the network model will have fault diagnosers for each of the OSI layers. The system includes generic layer 3 (routing) diagnosers, and specific layer 3 routing diagnosers, like an OSPF diagnoser to handle particular OSPF faults. The system also includes layer 2 diagnosers

(for subnets, LANS, switches, bridges, Ports, etc.), ATM diagnosers, and FrameRelay diagnosers. Each diagnoser understands the types of faults in the particular domain that it registers to handle and it knows how to determine the possible causes of a detected fault in its domain.

Furthermore, fault handler 150 may include diagnoser fault handlers 154 and tester fault handlers 152 designed to handle QoS problems, VPNs, Multicast Groups, Wireless connectivity, Cable access, DSL access, or other types of technology. As specifically described below, after a diagnoser finds the possible causes of a detected fault, tester fault handlers are invoked to determine whether the possible faults actually exist and record the test results. Thus, fault testers are strongly related to the fault diagnosers and to the models they query to test for problems. For each type of fault there may be several diagnoser fault handlers and several tester fault handlers.

Fault diagnosers know specific types of problems that may be causing a particular symptom reported in the detected fault. A diagnoser performs domain specific algorithms to determine possible causes. A diagnoser consults the objects that comprise the specific domain and gathers specific subcomponents by following the domain specific dependencies that have been established in the modeled system. A diagnoser may designate as a possible fault, every member of the domain. Alternatively, a diagnoser may generate a specific list of suspect members based on relationships to the parameters found in the input fault.

Ultimately, a tester performs tests on some subset of the objects in its domain. These tests involve access to the real world counterparts that they represent. Sometimes a subcomponent may represent an

abstract object (for example, the subnet object in the diagram above). These abstract objects may require further subcomponent analysis in order to determine the cause of the fault being diagnosed. This process can integrate seamlessly without disturbing any of the other existing diagnosers and testers.

A tester may include vendor and device-specific knowledge, which allows it to perform the tests needed to determine if a possible fault actually exists. Diagnosers ``know'' about components at a particular network layer and rely on the diagnosis capabilities of their constituent subcomponents to examine deeper through the layers as needed. As the system performs its analysis, a log is written to track the decision process made by the system as a whole including the results of the tests, which could be used by a network administrator to troubleshoot a network problem.

Fault detection and management system 100 utilizes a decomposition approach in the diagnosis of a fault to determine the root cause. Fault detector 130 enters a problem or potential problem into fault object factory 110, which creates a fault object treated as a symptom fault. The symptomatic fault is decomposed into one or more constituent faults that further refine the symptom as shown in Figs. 7 and 7A. Each constituent fault represents a possible suspect that may be causing the symptom. For each constituent fault, tests are performed to determine the existence of a problem, or the constituent fault is further decomposed into suspects. The process continues until all faults are completely decomposed and there are no more suspects.

The end result of this process is a hierarchy of faults in the form of a tree with the original symptomatic fault at the root (as shown in Figs. 7 and 7A). The fault tree includes a root fault level, one or several intermediate fault levels, and a leaf fault level. Each fault in the

tree, except the root, has at least one parent fault from which it was decomposed. Each fault also has zero or more child faults that were spawned from it by a diagnoser fault handler. A child fault represents a possible cause of its parent. A fault that has children but is not the root is termed an intermediate fault. A fault that has no children, that is one that could not be further decomposed, is termed a leaf fault. A leaf fault that indicates a problem is a probable cause of the root symptom. There may be more than one root cause.

Fault diagnosis engine 101 computes a composite result for two or more child faults according to the following default rules for composite result computation:

1. If any child fault result state is PROBLEM, then the parent fault's result state is PROBLEM;
2. If all child fault result states are NO_PROBLEM, then the parent fault's result is NO_PROBLEM; and
3. Otherwise, the parent fault's result is UNKNOWN.

Using the above rules, the composite result for a parent fault is entered.

Figs. 6A through 6F illustrate path dependent fault diagnosis in a communications network shown in Fig. 6. Referring to Figs. 6 and 6A, an agent A in subnet 22 (IP address 9.7.4.12) notifies a network management system (NMS) that it can no longer communicate at all with a server in subnet 28 at IP address 6.2.1.5. (step 202). As notification source, agent A uses an SNMP trap (or another notification method). The NMS translates agent's notification to a CommunicationLost fault (fault 302 in Fig. 7), specifying the agent's IP address as the source, and the server's IP as the destination (step 204). Next, in Step 206, a CommunicationLost fault diagnoser fault handler (CLFD) is registered for CLF faults, and gets triggered. Importantly, a topology source provides a Layer 3 path from the source to the destination (step 208). There are various topology sources known in the prior art. Furthermore, a

suitable topology source is also described in a co-pending PCT application entitled ``Systems and Methods for Constructing Multi-Layer Topological Models of Computer Networks'' (Docket No. A3-02WO) filed on 7 May 2001, which is incorporated by reference..

The topology source (e.g. topology mapper 14) returns an L3 path that usually includes a ``list of paths''. Each path is an ordered list of connections from a source device to a destination device specifying each device hop along the way and the port level connectivity. If the Layer 3 path cannot be determined, the CLF diagnoser does nothing (step 211).

Referring to Fig. 6B, if the Layer 3 path can be determined, then the CLF diagnoser fault handler creates a possible RouterConfiguration fault (RCF) on the router (step 216). This is done for each router in each Layer 3 path between the source and the destination. The RouterConfiguration fault indicates a condition in which the router is preventing traffic from being transmitted from a first device to a second device or vice-versa.

In step 218, for each connection in each Layer 3 path between the source and the destination, the CLF diagnoser fault handler creates a possible CommunicationLostSubnet fault (CLSF) between port 1 of device 1 and port 2 of device 2 (step 220). The CommunicationLostSubnet faults indicate a condition in which traffic cannot be transmitted from device 1 to device 2 within a given subnet. In step 222, fault diagnosis engine 101 registers both a CLSF Tester fault handler (CLSFT) and CLSF Diagnoser fault handler (CLSFD) to receive CLS faults. The CLSF Tester is triggered first by the fault diagnosis engine. CLSFT requests that device 1 do a remote ping of device 2 (or vice-versa, or both) (step 230). The connection list returned from topology mapper 14 includes information about the devices in the connection. The information includes an identity of the type of the device (i.e., SysDID), and includes the RemotePing

Capability (RPC), which is the ability of a device to attempt to contact another device via ICMP and report back the result.

If device 1 or device 2 do not support remote pings, CLSF tester adds a test result of UNKNOWN to the CLSF (step 226). Then, fault diagnosis engine 101 invokes the CLSF diagnoser on the CLS fault (step 228). If devices 1 and 2 support remote pings, the CLSF tester sets up a remote ping request. Then, the CLSF tester receives a remote ping result from device 1 and 2 (step 232), and determines if the remote ping was successful (step 234).

In step 238, if the remote ping was not successful, the CLSF tester adds test result of PROBLEM to the CLS fault. Then, fault engine 101 sets the CLS fault to the detected state (step 240). Next, fault diagnosis engine 101 invokes the CLSF diagnoser on the CLS fault (step 242). If the remote ping was successful, the CLSF tester adds test result of NO_PROBLEM to the CLS fault (step 244). Next, fault diagnosis engine 101 sets the CLS fault to the completed state (step 246). In this case, the CLSF diagnoser never gets triggered because test results shows the CLS fault does not exist in this subnet (step 250).

The CLSF diagnoser uses the topology source in the NMS to request a Layer 2 path from device 1 to device 2 (step 252). If Layer 2 path cannot be determined, the CLSF diagnoser does nothing (step 256). Then, fault diagnosis engine 101 triggers other handlers that are registered for this type of fault (step 258). If layer 2 path can be determined for each connection in the Layer 2 path, between port 1 of device 1 and port 2 of device 2, the CLSF diagnoser creates a possible ConnectionDownFault (CDF) between port 1 and port 2 (step 262). The ConnectionDown fault (see faults 314, 316, and 318 in Fig. 7A) is a fault indicative of a condition in which the traffic cannot be transmitted between the two ports of the corresponding devices 1 and 2.

Referring to Fig. 6D, if the CDF diagnoser (CDFD) is registered for the CDF faults, fault diagnosis engine 101 triggers the CDF diagnoser (step 264). The CDF diagnoser creates two subordinate PortDownFaults (PDF) (see, e.g., 320 and 322 in Fig. 7A), one for each port in the CDF (step 266).

If the PDF tester (PDFT) is registered for PDF Faults, fault diagnosis engine 101 triggers the PDF tester (step 268). The PDF tester then determines the type of test to run to determine existence of the fault (step 270). If the PDF tester is able to determine the type of test, the PDF tester performs test to check status of port (step 273). For example, if the port has a manifestation in the MIB-II ifTable of the device's SNMP agent, the PDFT would read the ifOperStatus of this port, and compare the value to a set of known good values (generally, ON (1) is the only good value). If the port's ifOperStatus value is included in the set of good values for ifOperStatus, the result state 125 of the created test result object 120 is NO_PROBLEM; otherwise, it is PROBLEM. Regardless of the test state, the actual value of the ifOperStatus is recorded in the test data 124.

If the PDF tester cannot determine the type of test, the PDF tester adds a test result of UNKNOWN to the PDF (step 274), as shown by fault 328 in Fig. 7A. Then, fault diagnosis engine 101 triggers other listeners that are registered for this type of fault (step 276).

Referring to Fig. 6E, if the PDF tester determines that the port was operationally up, the PDF tester sets the PDF fault to NO_PROBLEM (step 281), and fault diagnosis engine 101 sets the PDF to the completed state. After setting PDF to the completed state, the diagnosis of the fault is complete. Fault diagnosis engine 101 does not trigger any other handlers for this fault (step 283).

On the other hand, if the PDF tester determines that the port was down (step 277), the PDF tester adds, in step 281, test result of PROBLEM to the PDF fault (see, e.g., 324

of Fig. 7A). Then, fault diagnosis engine 101 sets the PDF to be the detected state. Next, fault diagnosis engine 101 triggers other diagnosers that are registered for this type of fault (i.e., PDF).

Fig. 6F shows steps taken regarding a possible RouterConfiguration fault. If an RCF tester (RCFT) is registered to receive RDF faults, fault diagnosis engine 101 triggers the RCF tester (step 284). Next, the RCF tester determines the type of the device and the proper tests to perform to test existence of a router configuration problem (step 285). For example, if the router's SNMP agent supports a route forwarding database, such as the MIB-II ipRouteTable or the ipCidrTable (defined in rfc 2096), the forwarding database would be queried to see if a static route had been configured causing the traffic flow to deviate from the intended flow according to the routing protocol(s) running on the router. Other configuration and implementation errors can also be detected. Any condition detected on the router that could adversely impact the forwarding of data from the source to the destination, will cause the test result for the RouterConfiguration fault to have a result state of PROBLEM. If the RCF tester is not able to determine proper tests to perform for testing router configuration (step 286), then the RCF tester adds a test result of UNKNOWN to the RCF fault (step 288). Fault diagnosis engine 101 triggers other listeners registered for this type of fault (step 290).

If the RCF tester is able to determine the type of the device and the proper tests to perform to test existence of a router configuration problem, the RCF tester performs these configuration tests (step 292). The tests determine whether the tested router is misconfigured (step 293). If the tested router is misconfigured, the RCF tester adds test result of PROBLEM to the RCF fault (step 294). Next, fault diagnosis engine 101 sets the RCF to the detected state (step 295). Then, fault diagnosis engine 101 triggers other

diagnosers registered for this type of fault (step 296).

If the RCF tester determines that the tested router is not misconfigured (step 293), the RCF tester adds test result of NO_PROBLEM to the RCF fault (step 297). Next, fault diagnosis engine 101 sets the RCF to the completed state (step 298). Then, fault diagnosis engine 101 triggers other listeners registered for this type of fault (step 299).

Numerous other embodiments not described in detail here can apply the principles described to particular applications and are within the scope of the claims.

What is claimed is:

CLAIMS

1. A method of diagnosing a fault in a communications network, comprising the acts of:

receiving fault data;

determining path in the communications network related to said fault data; and

performing, using said path, a fault analysis on said fault data to diagnose a fault related to the communications network.

2. The method of claim 1 wherein said a fault analysis is a root cause analysis.

3. The method of claim 1 or 2 wherein said determining path includes performing a domain-specific algorithm.

4. The method of claim 3 wherein said path determination includes dynamically generating a trace route by employing a path-tracing algorithm.

5. The method of claim 1 or 2 wherein said determining path includes employing data stored in a topology mapper.

6. The method of claim 1 or 2 wherein said determining path includes using routing and switching algorithms used by the network for data forwarding.

7. The method of claim 1 or 2 wherein said determining path includes performing a layer 3 path determination.

8. The method of claim 7 wherein said performing said layer 3 path determination includes finding a router in a subnet and using said router as a starting point for path generation.

9. The method of claim 2 wherein said receiving fault

data includes creating a fault object in a fault object factory using said fault data.

10. The method of claim 2 wherein said receiving fault data includes using a detector.

11. The method of claim 2 wherein said using a detector includes using an agent.

12. The method of claim 2 wherein said receiving fault data includes creating a fault object.

13. The method of claim 12 wherein said performing root cause analysis include triggering a specific fault handler.

14. The method of claim 13 wherein said triggering said fault handler includes employing a diagnoser fault handler.

15. The method of claim 14 wherein said employing said diagnoser fault handler includes generating possible faults that may caused occurrence of said fault data.

16. The method of claim 13 wherein said triggering said fault handler includes employing a tester fault handler.

17. The method of claim 15 wherein said triggering said fault handler includes employing a tester fault handler of each said possible fault.

18. The method of claim 13 wherein said employing fault handler includes obtaining an ordered list of fault handlers for a specified transition state of a fault object.

19. The method of claim 18 wherein said obtaining said ordered list includes employing a diagnoser fault handler registered for the type of the analyzed object.

20. The method of claim 14 wherein said diagnoser fault handler is designed to handle QoS problems.

21. The method of claim 14 wherein said diagnoser fault handler is designed to handle problems in virtual private networks.

22. The method of claim 14 wherein said diagnoser fault handler is designed to handle problems in multi-cast groups.

23. The method of claim 14 wherein said diagnoser fault handler is designed to handle wireless connectivity problems.

24. The method of claim 14 wherein said diagnoser fault handler is designed to handle problems related to cable access.

25. The method of claim 14 wherein said diagnoser fault handler is designed to handle problems related to DSL access.

26. The method of claim 5 wherein said employing diagnoser fault handler includes transitioning fault object between processing states.

27. The method of claim 2 further including prioritization.

28. The method of claim 2 further including fault presentation that displays fault result to a user.

29. A system for diagnosing a fault in a communications network, comprising:

a fault object factory constructed and arranged to receive fault data and create fault objects; and

a path determination module constructed to determine, in the communications network, a path related to said fault data; and

a fault diagnosis engine constructed to perform, using said path, a fault analysis on said fault data to diagnose a fault related to the communications network.

30. The system of claim 29 further including fault detector constructed and arranged to detect a fault in a monitored entity.

31. The system of claim 29 further including a fault repository constructed and arranged for storing and accessing fault objects.

32. The system of claim 29 further including a fault handler constructed to perform analysis of said fault object.

33. The system of claim 32 wherein said fault handler includes a fault handler tester.

34. The system of claim 32 wherein that fault handler includes a diagnoser fault handler constructed and arranged to change a fault state of said object.

35. The system of claim 31 wherein said fault repository include said fault object and a fault association object.

36. The system of claim 29 wherein said fault object

includes a description, a processing state, and test result object.

37. The system of claim 36 wherein said test result object includes a description, a target entity, test data, and a result state.

38. A network management system, comprising a fault diagnosis system connectable to a communications network, comprising:

a fault object factory constructed and arranged to receive fault data and create fault objects; and

a path determination module constructed to determine, in the communications network, a path related to said fault data; and

a fault diagnosis engine constructed to perform, using said path, a fault analysis on said fault data to diagnose a fault related to the communications network.

39. The network management system of claim 38 further comprising a help desk system.

40. The network management system of claim 38 further comprising a topology mapper.

41. The network management system of claim 38 further comprising an impact analyzer.

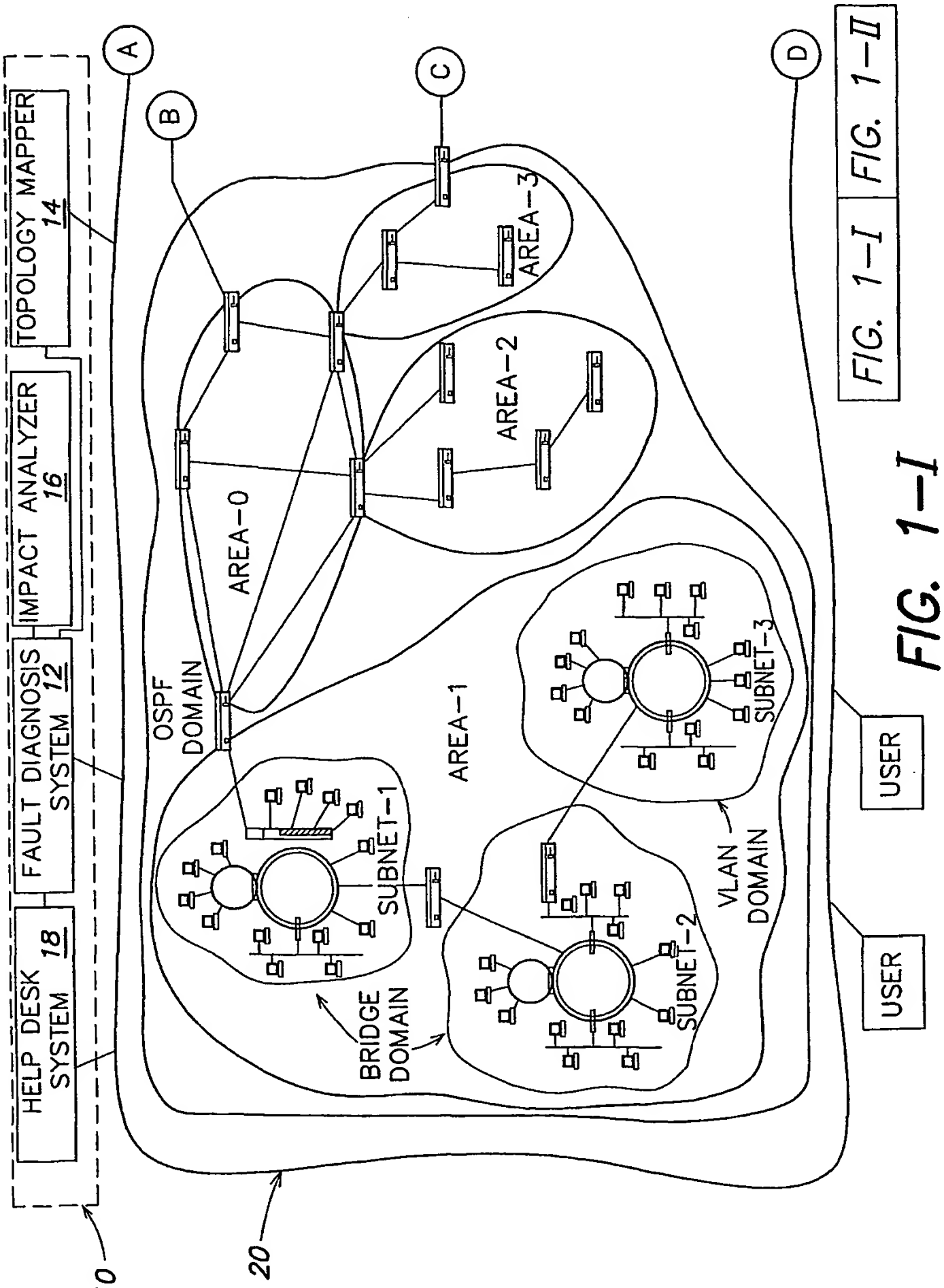


FIG. 1-I | FIG. 1-II

FIG. 1-I

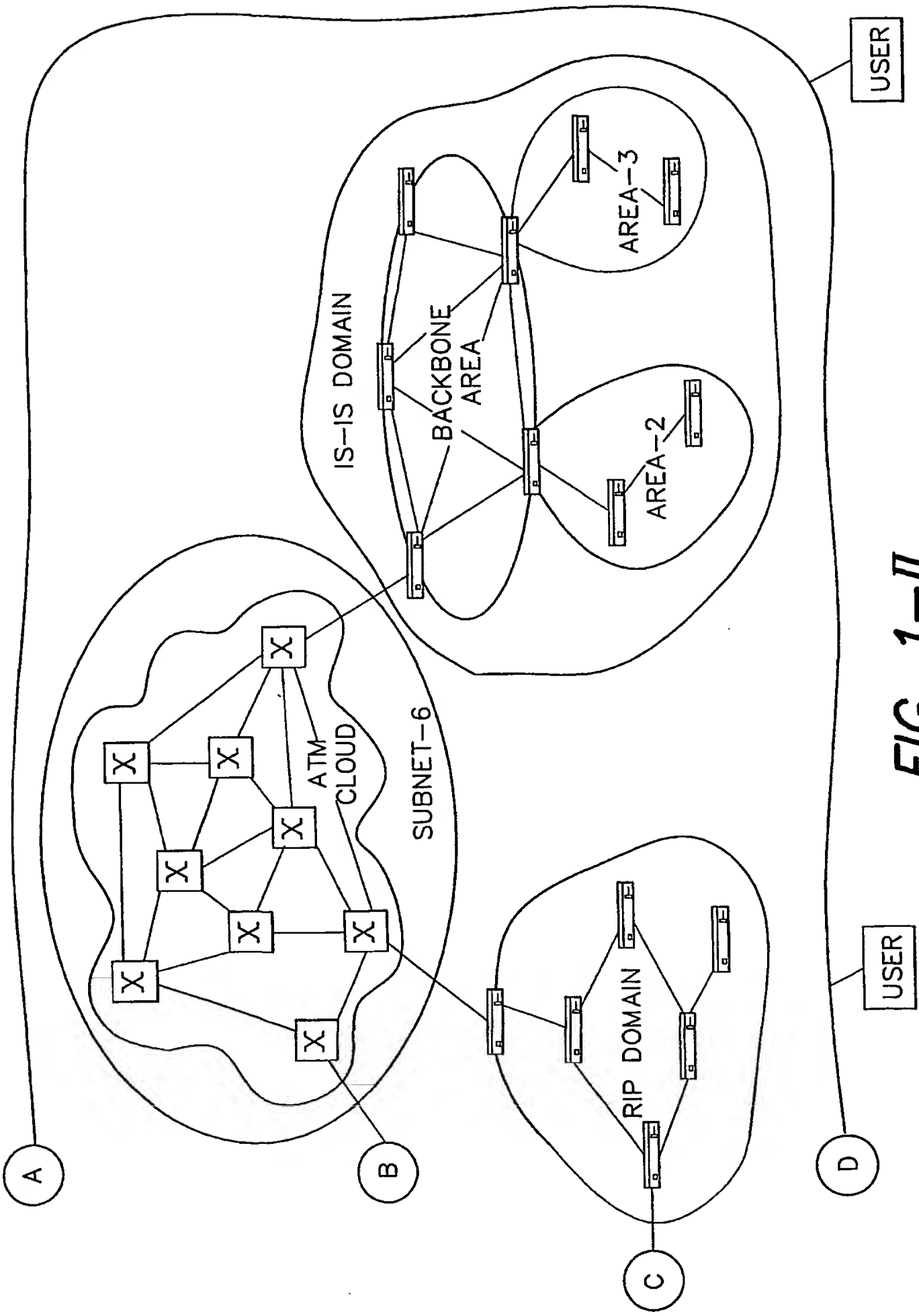

USER
IS–IS DOMAIN
BACKBONE AREA
AREA–3
AREA–2
SUBNET–6
ATM CLOUD
RIP DOMAIN
USER
A
B
C
D

FIG. 1–II

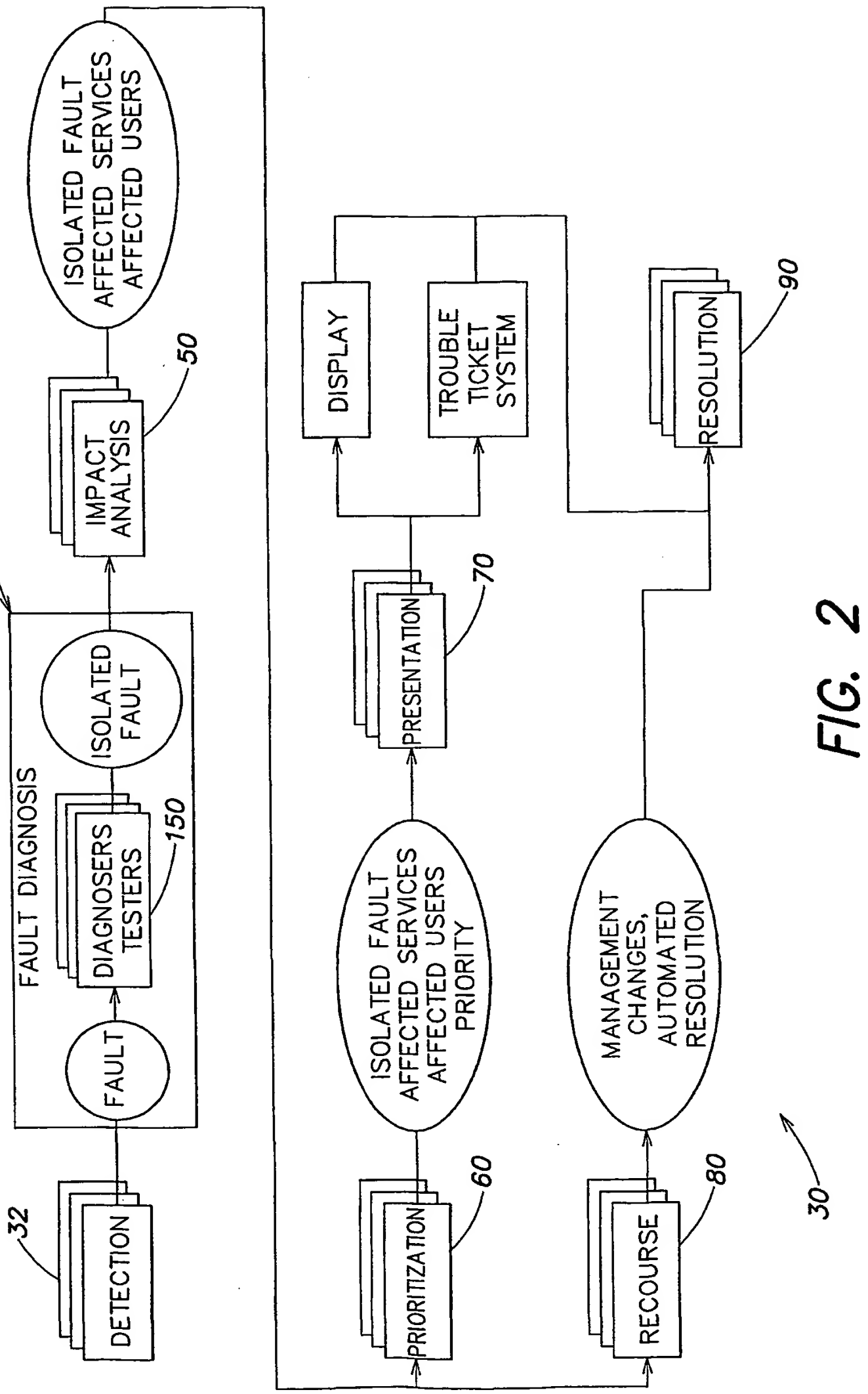

FAULT DIAGNOSIS
DETECTION
32
FAULT
ISOLATED FAULT
DIAGNOSERS TESTERS
150
IMPACT ANALYSIS
50
ISOLATED FAULT AFFECTED SERVICES AFFECTED USERS
40
PRIORITIZATION
60
ISOLATED FAULT AFFECTED SERVICES AFFECTED USERS PRIORITY
PRESENTATION
70
DISPLAY
TROUBLE TICKET SYSTEM
RECOURSE
80
MANAGEMENT CHANGES, AUTOMATED RESOLUTION
RESOLUTION
90
30

FIG. 2

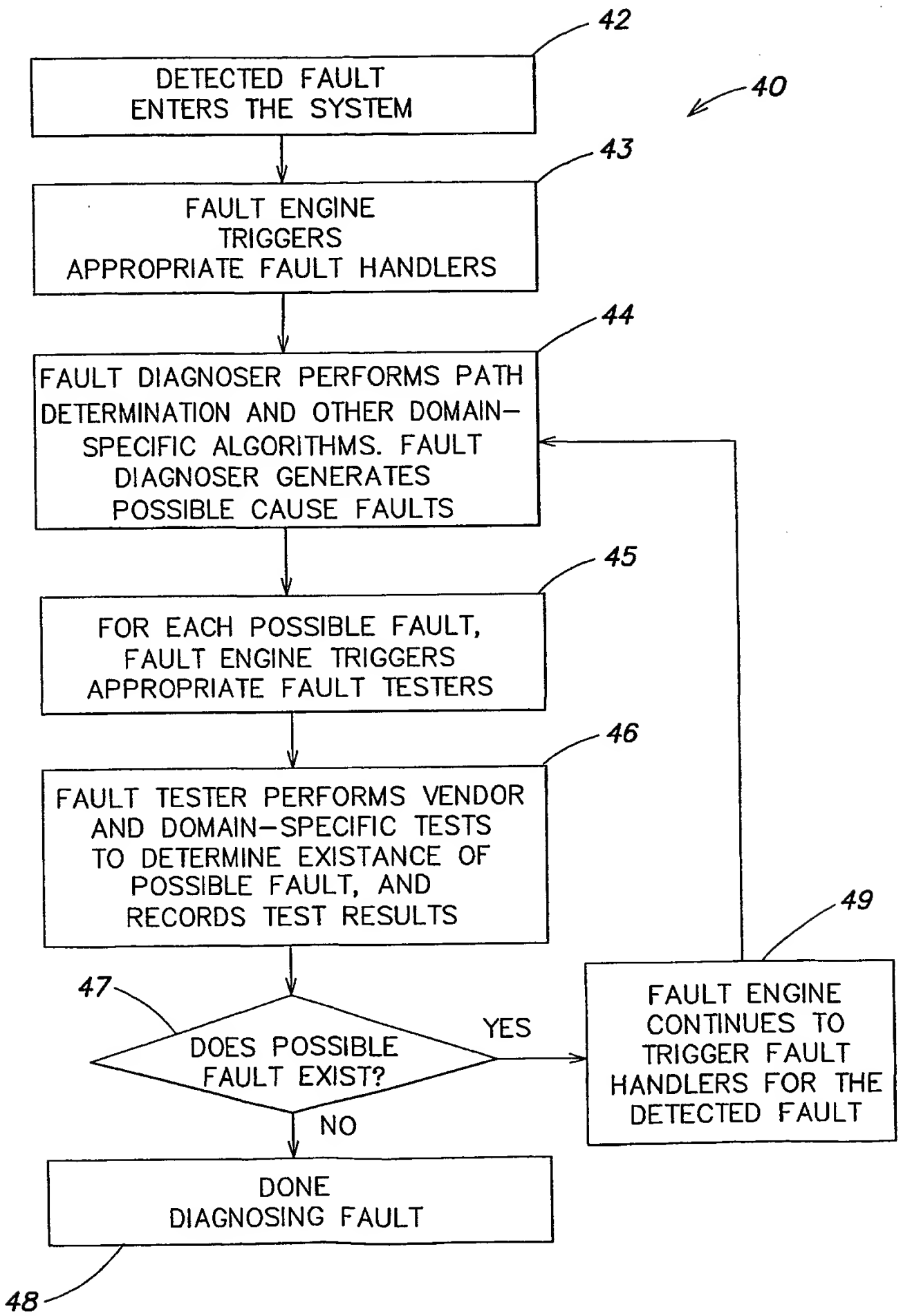

42
DETECTED FAULT ENTERS THE SYSTEM
40
43
FAULT ENGINE TRIGGERS APPROPRIATE FAULT HANDLERS
44
FAULT DIAGNOSER PERFORMS PATH DETERMINATION AND OTHER DOMAIN-SPECIFIC ALGORITHMS. FAULT DIAGNOSER GENERATES POSSIBLE CAUSE FAULTS
45
FOR EACH POSSIBLE FAULT, FAULT ENGINE TRIGGERS APPROPRIATE FAULT TESTERS
46
FAULT TESTER PERFORMS VENDOR AND DOMAIN-SPECIFIC TESTS TO DETERMINE EXISTANCE OF POSSIBLE FAULT, AND RECORDS TEST RESULTS
47
DOES POSSIBLE FAULT EXIST?
YES
NO
49
FAULT ENGINE CONTINUES TO TRIGGER FAULT HANDLERS FOR THE DETECTED FAULT
48
DONE DIAGNOSING FAULT

FIG. 2A

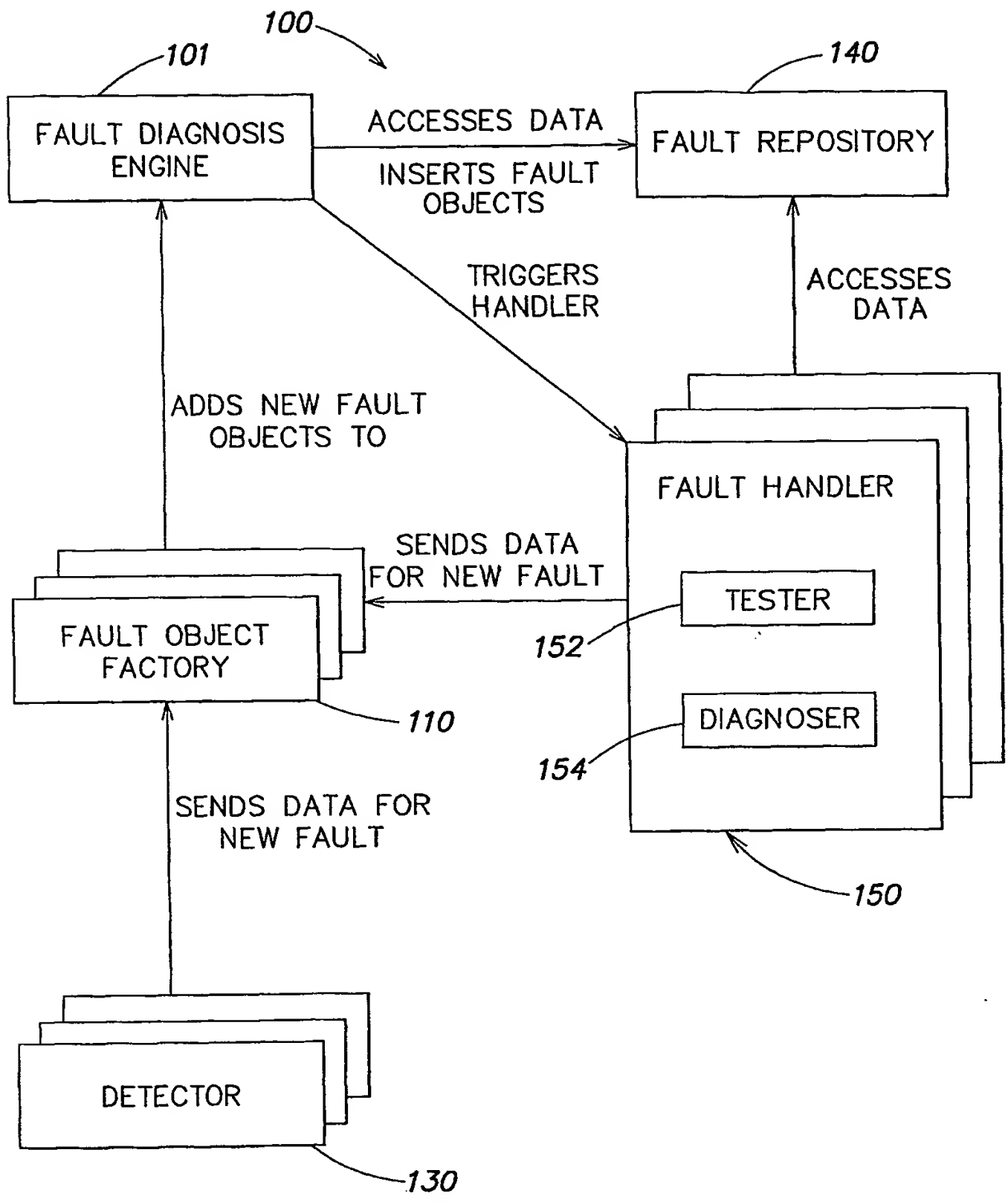

100
101
140
FAULT DIAGNOSIS ENGINE
ACCESSES DATA
INSERTS FAULT OBJECTS
FAULT REPOSITORY
TRIGGERS HANDLER
ACCESSES DATA
ADDS NEW FAULT OBJECTS TO
FAULT HANDLER
SENDS DATA FOR NEW FAULT
TESTER
152
FAULT OBJECT FACTORY
110
DIAGNOSER
154
SENDS DATA FOR NEW FAULT
150
DETECTOR
130

FIG. 3

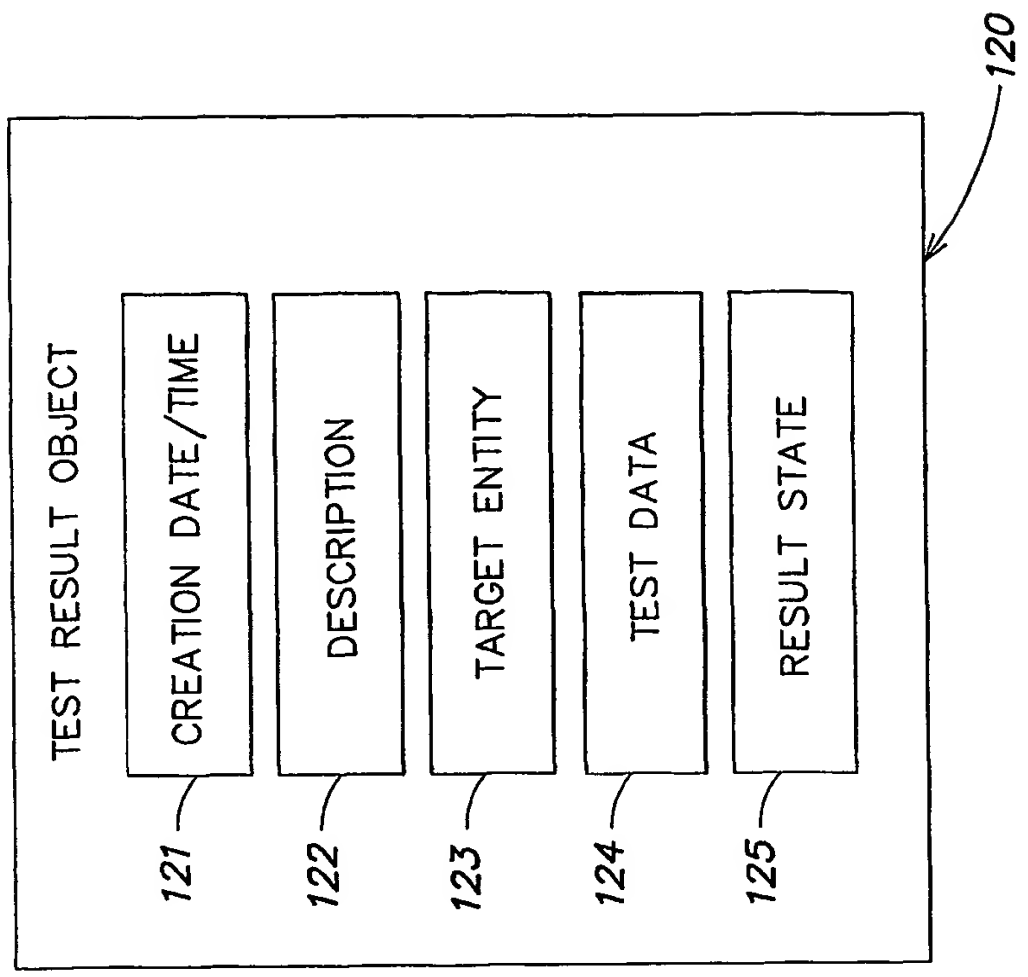

TEST RESULT OBJECT
CREATION DATE/TIME
DESCRIPTION
TARGET ENTITY
TEST DATA
RESULT STATE
121
122
123
124
125
120

FIG. 3C

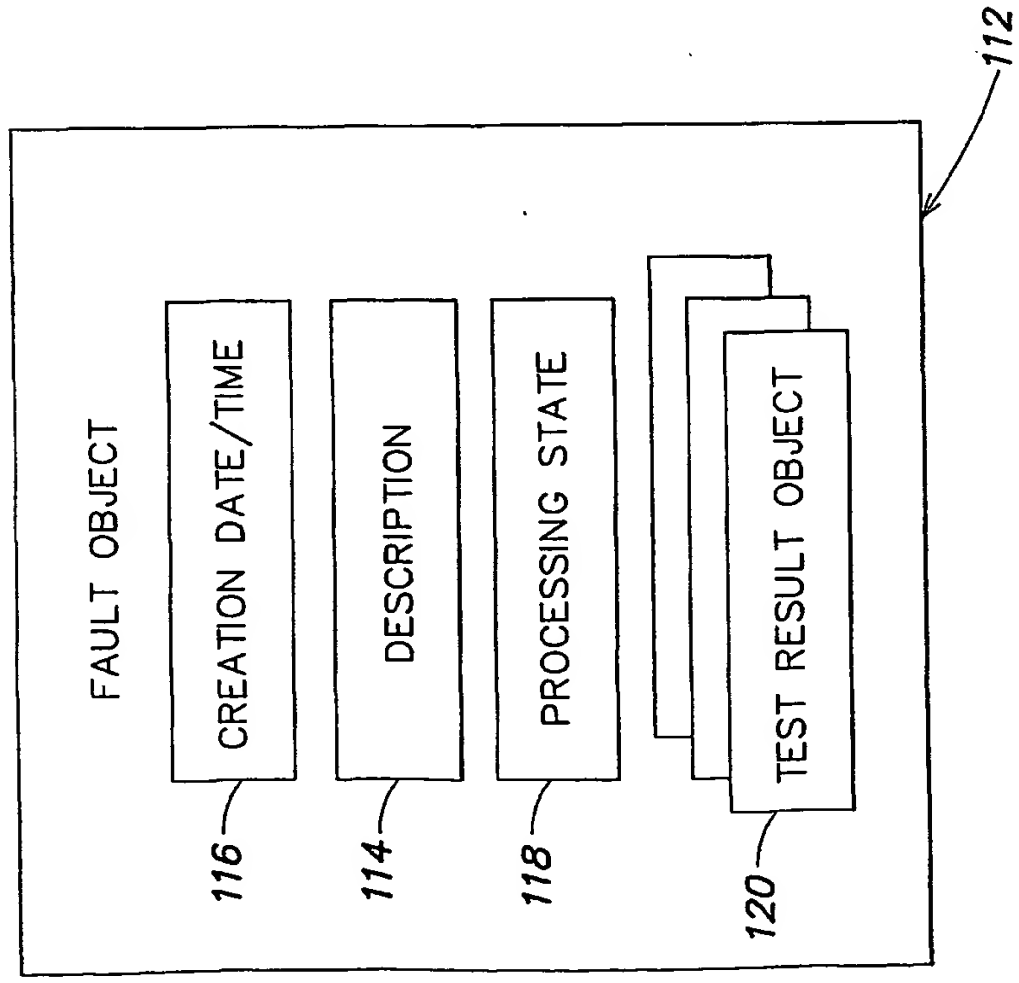

FAULT OBJECT
CREATION DATE/TIME
DESCRIPTION
PROCESSING STATE
TEST RESULT OBJECT
116
114
118
120
112

FIG. 3A

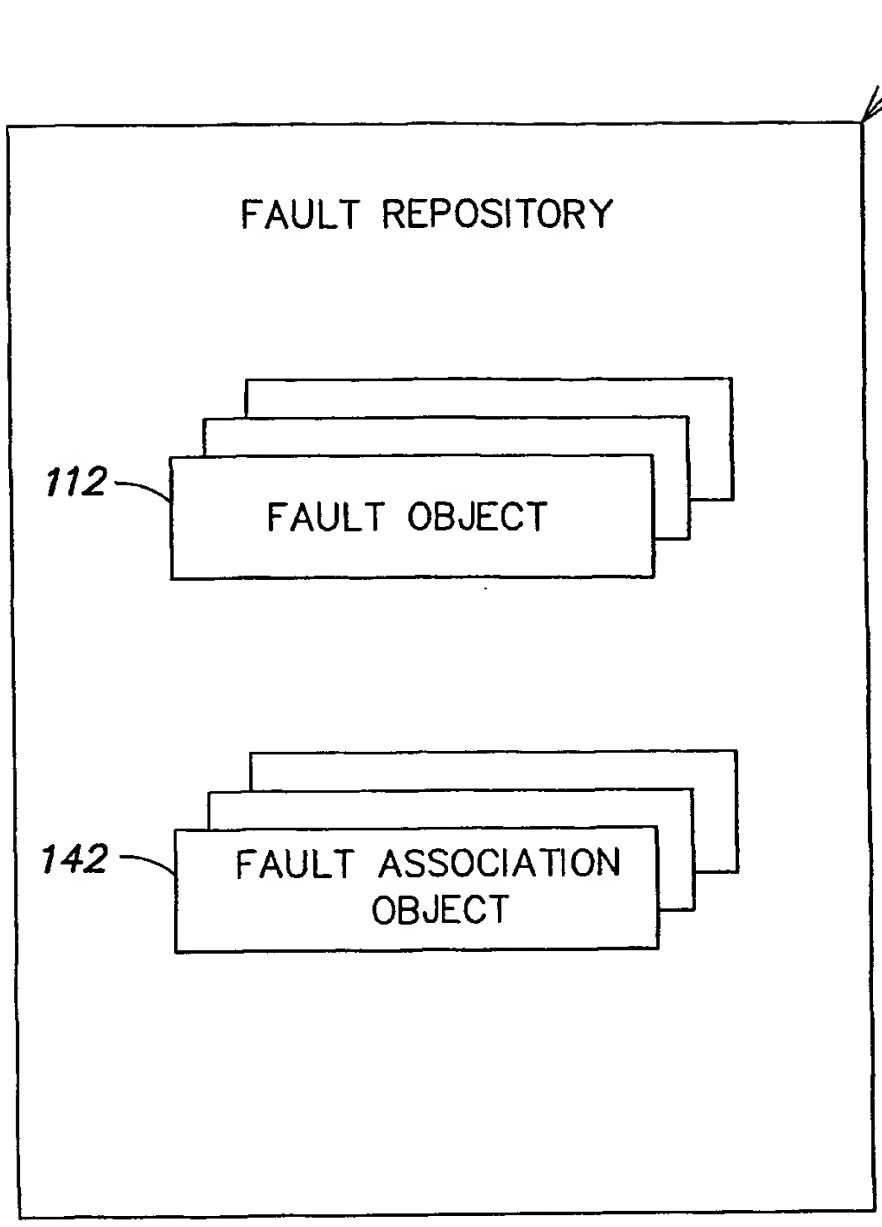

140
FAULT REPOSITORY
112
FAULT OBJECT
142
FAULT ASSOCIATION OBJECT

FIG. 3B

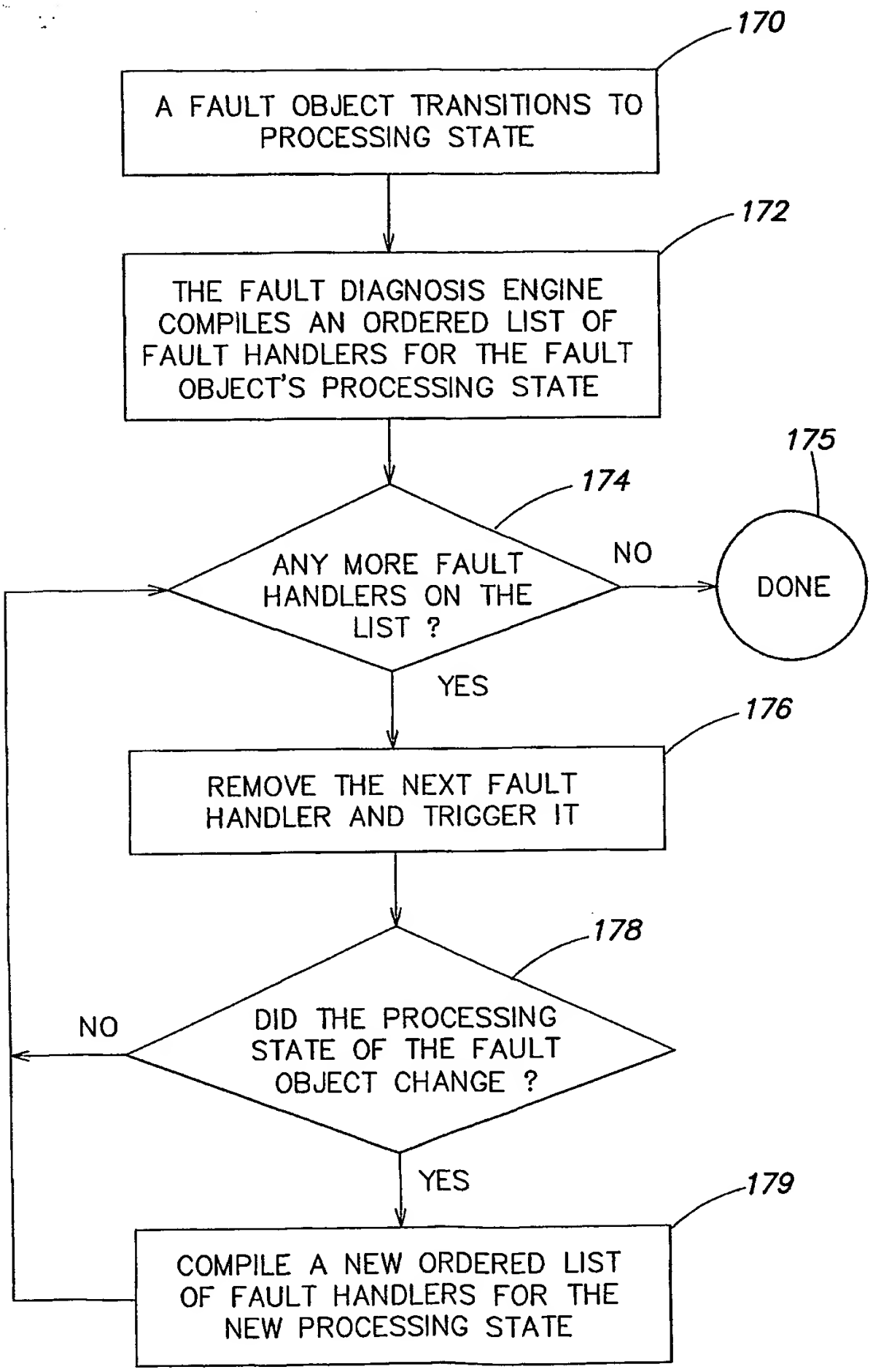

170
A FAULT OBJECT TRANSITIONS TO PROCESSING STATE
172
THE FAULT DIAGNOSIS ENGINE COMPILES AN ORDERED LIST OF FAULT HANDLERS FOR THE FAULT OBJECT'S PROCESSING STATE
175
174
ANY MORE FAULT HANDLERS ON THE LIST ?
NO
DONE
YES
176
REMOVE THE NEXT FAULT HANDLER AND TRIGGER IT
178
DID THE PROCESSING STATE OF THE FAULT OBJECT CHANGE ?
NO
YES
179
COMPILE A NEW ORDERED LIST OF FAULT HANDLERS FOR THE NEW PROCESSING STATE

FIG. 4

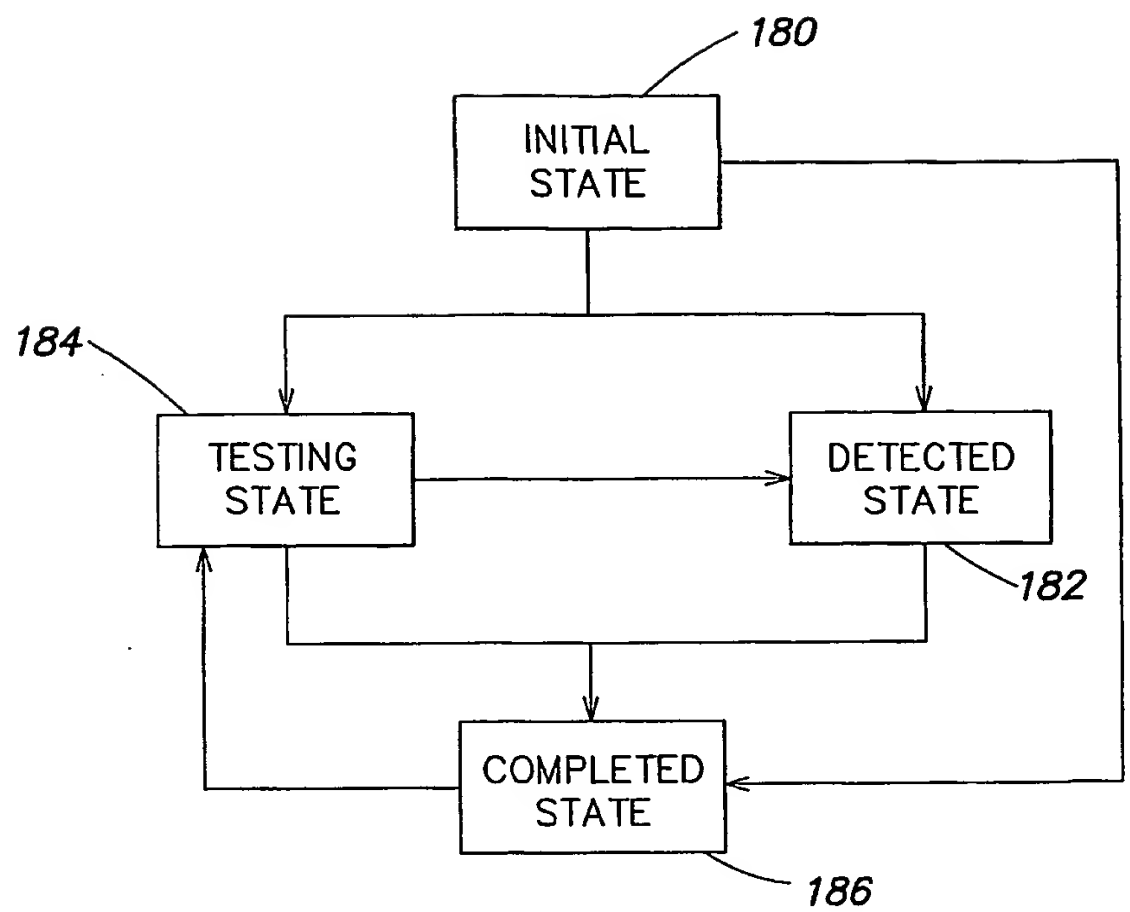

180
INITIAL STATE
184
TESTING STATE
DETECTED STATE
182
COMPLETED STATE
186

FIG. 5

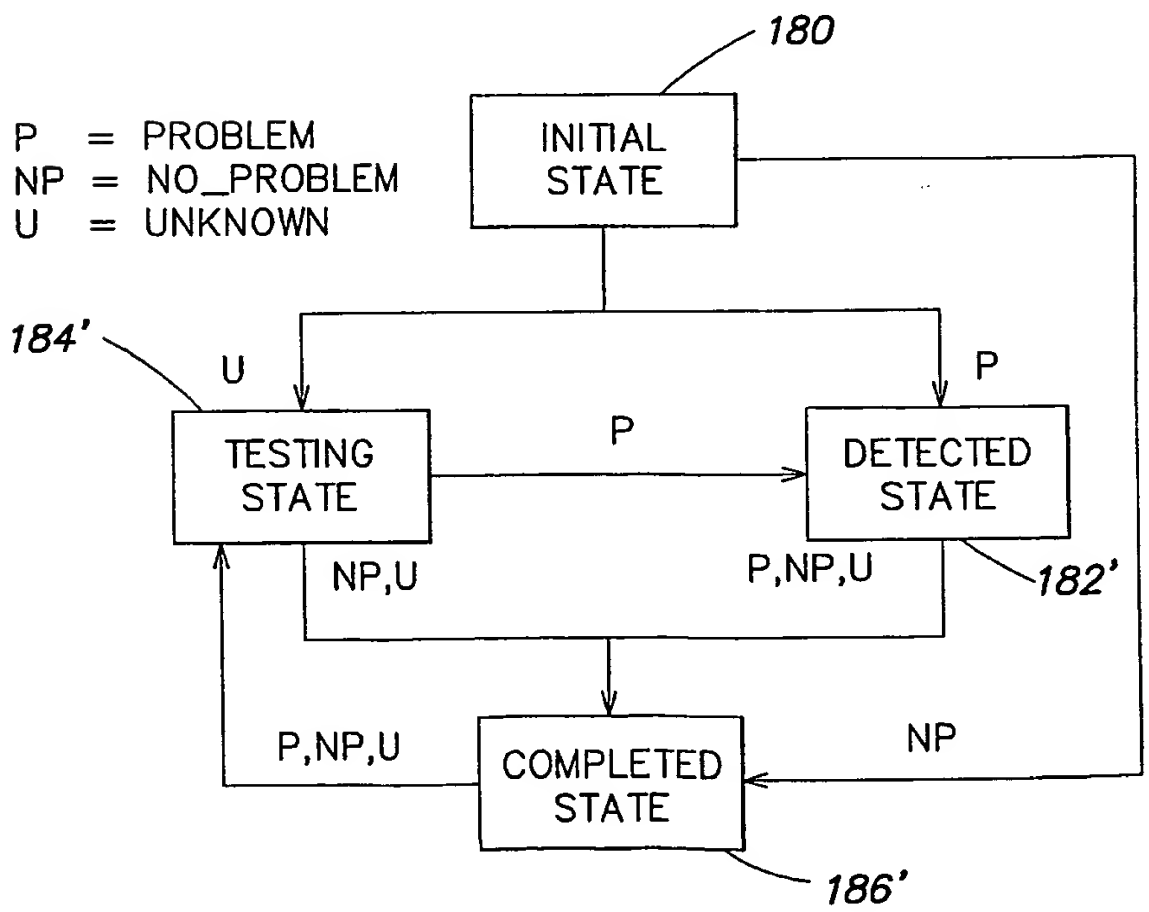

P = PROBLEM
NP = NO_PROBLEM
U = UNKNOWN
180
INITIAL STATE
184'
U
P
TESTING STATE
P
DETECTED STATE
NP,U
P,NP,U
182'
P,NP,U
COMPLETED STATE
NP
186'

FIG. 5A

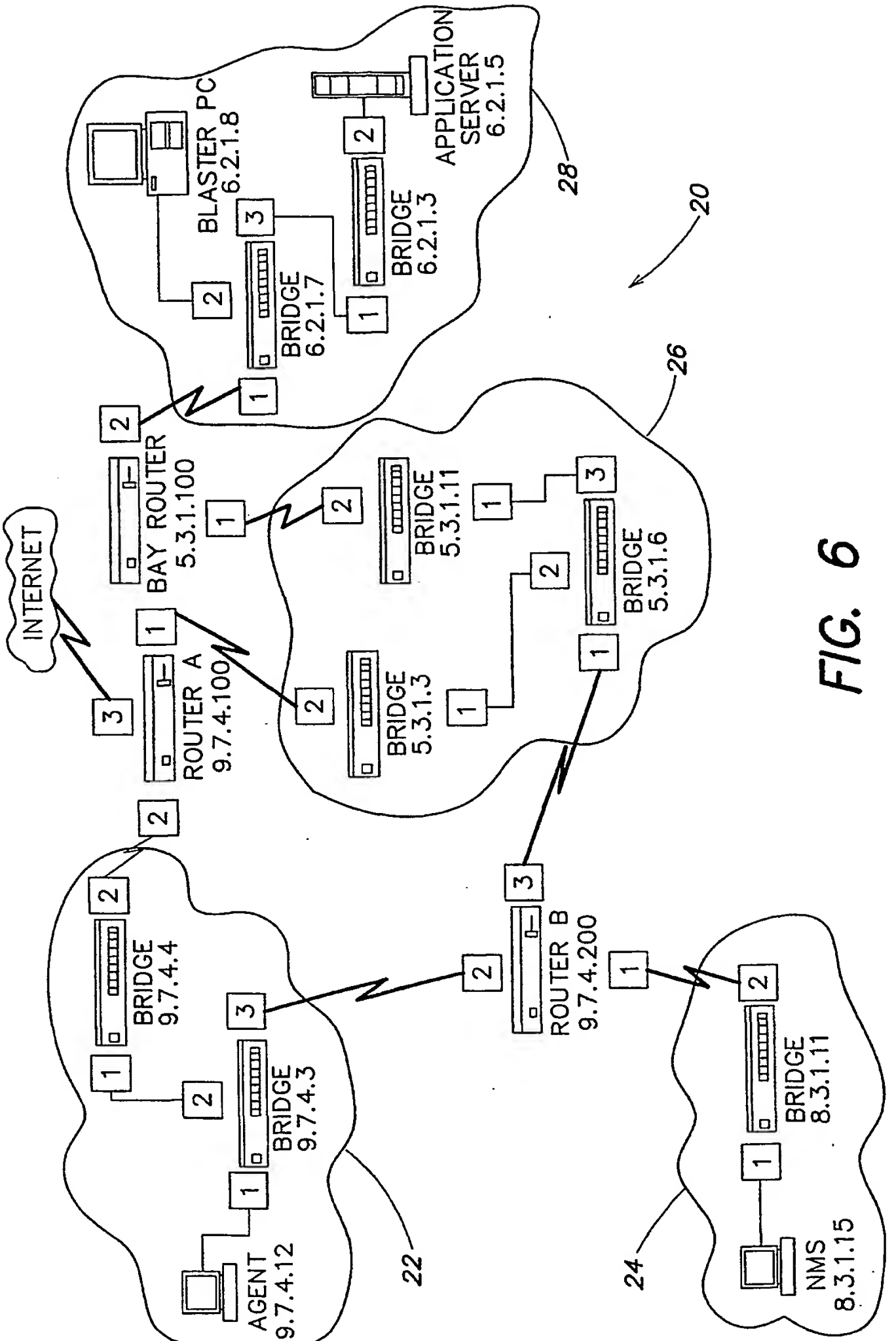

INTERNET
BAY ROUTER
5.3.1.100
ROUTER A
9.7.4.100
ROUTER B
9.7.4.200
BLASTER PC
6.2.1.8
BRIDGE
6.2.1.7
BRIDGE
6.2.1.3
APPLICATION
SERVER
6.2.1.5
BRIDGE
5.3.1.11
BRIDGE
5.3.1.3
BRIDGE
5.3.1.6
BRIDGE
9.7.4.4
BRIDGE
9.7.4.3
AGENT
9.7.4.12
BRIDGE
8.3.1.11
NMS
8.3.1.15
20
22
24
26
28

FIG. 6

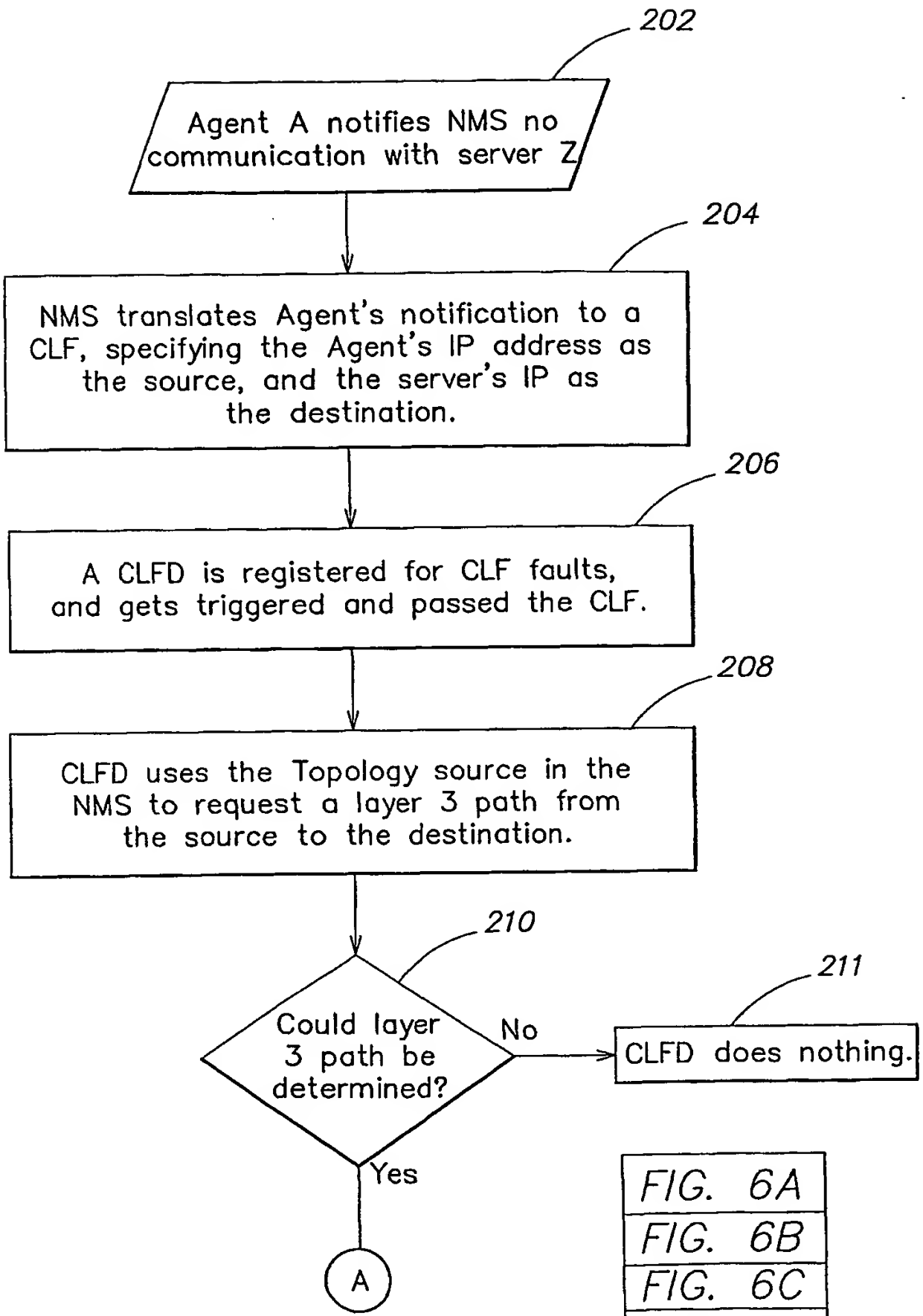

202
Agent A notifies NMS no communication with server Z
204
NMS translates Agent's notification to a CLF, specifying the Agent's IP address as the source, and the server's IP as the destination.
206
A CLFD is registered for CLF faults, and gets triggered and passed the CLF.
208
CLFD uses the Topology source in the NMS to request a layer 3 path from the source to the destination.
210
Could layer 3 path be determined?
No
211
CLFD does nothing.
Yes
A
FIG. 6A
FIG. 6B
FIG. 6C
FIG. 6D
FIG. 6E
FIG. 6F

FIG. 6A

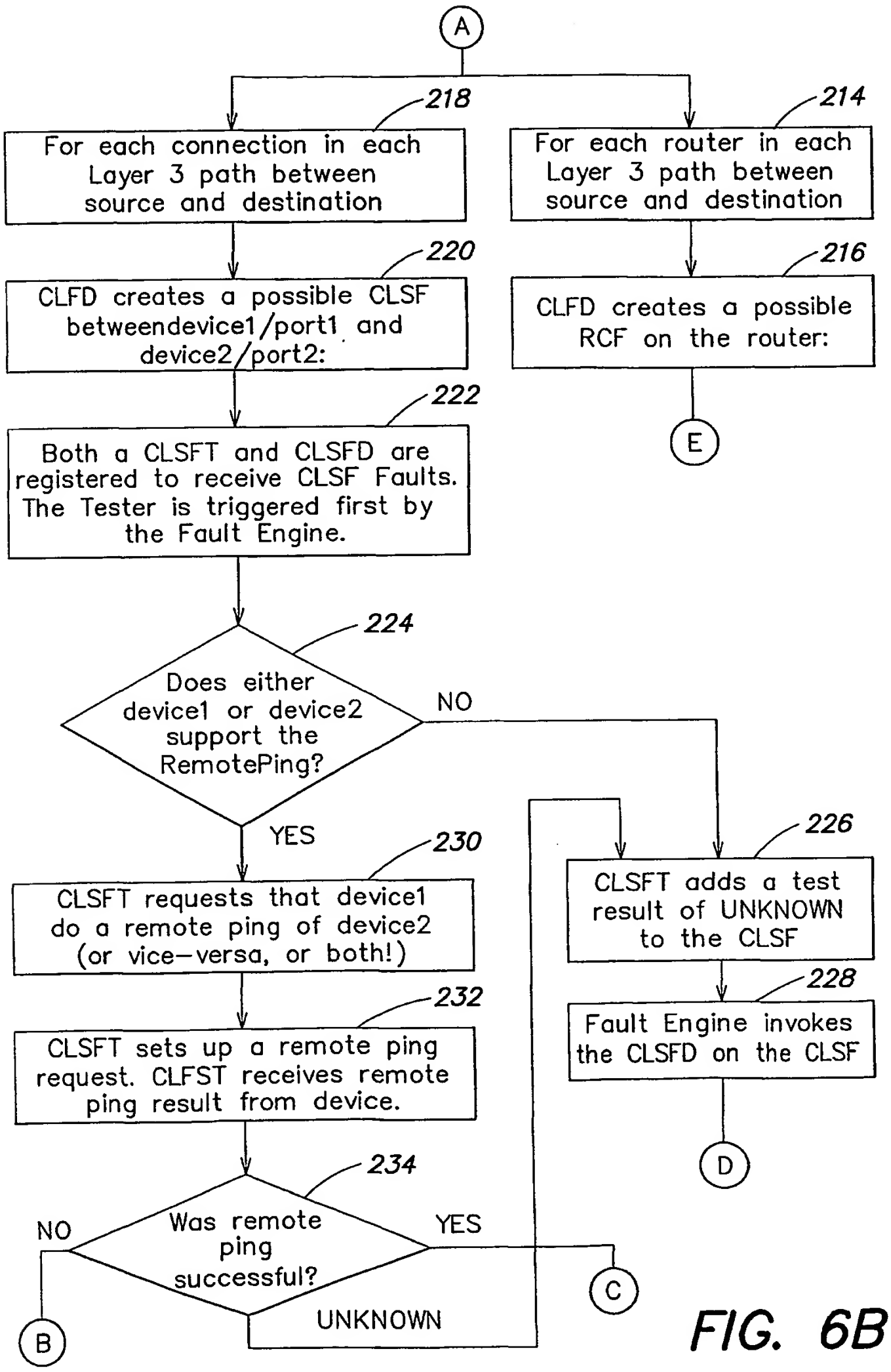

A
218
For each connection in each Layer 3 path between source and destination
220
CLFD creates a possible CLSF betweendevice1/port1 and device2/port2:
222
Both a CLSFT and CLSFD are registered to receive CLSF Faults. The Tester is triggered first by the Fault Engine.
224
Does either device1 or device2 support the RemotePing?
NO
YES
230
CLSFT requests that device1 do a remote ping of device2 (or vice-versa, or both!)
232
CLSFT sets up a remote ping request. CLFST receives remote ping result from device.
234
Was remote ping successful?
NO
YES
UNKNOWN
B
C
214
For each router in each Layer 3 path between source and destination
216
CLFD creates a possible RCF on the router:
E
226
CLSFT adds a test result of UNKNOWN to the CLSF
228
Fault Engine invokes the CLSFD on the CLSF
D

FIG. 6B

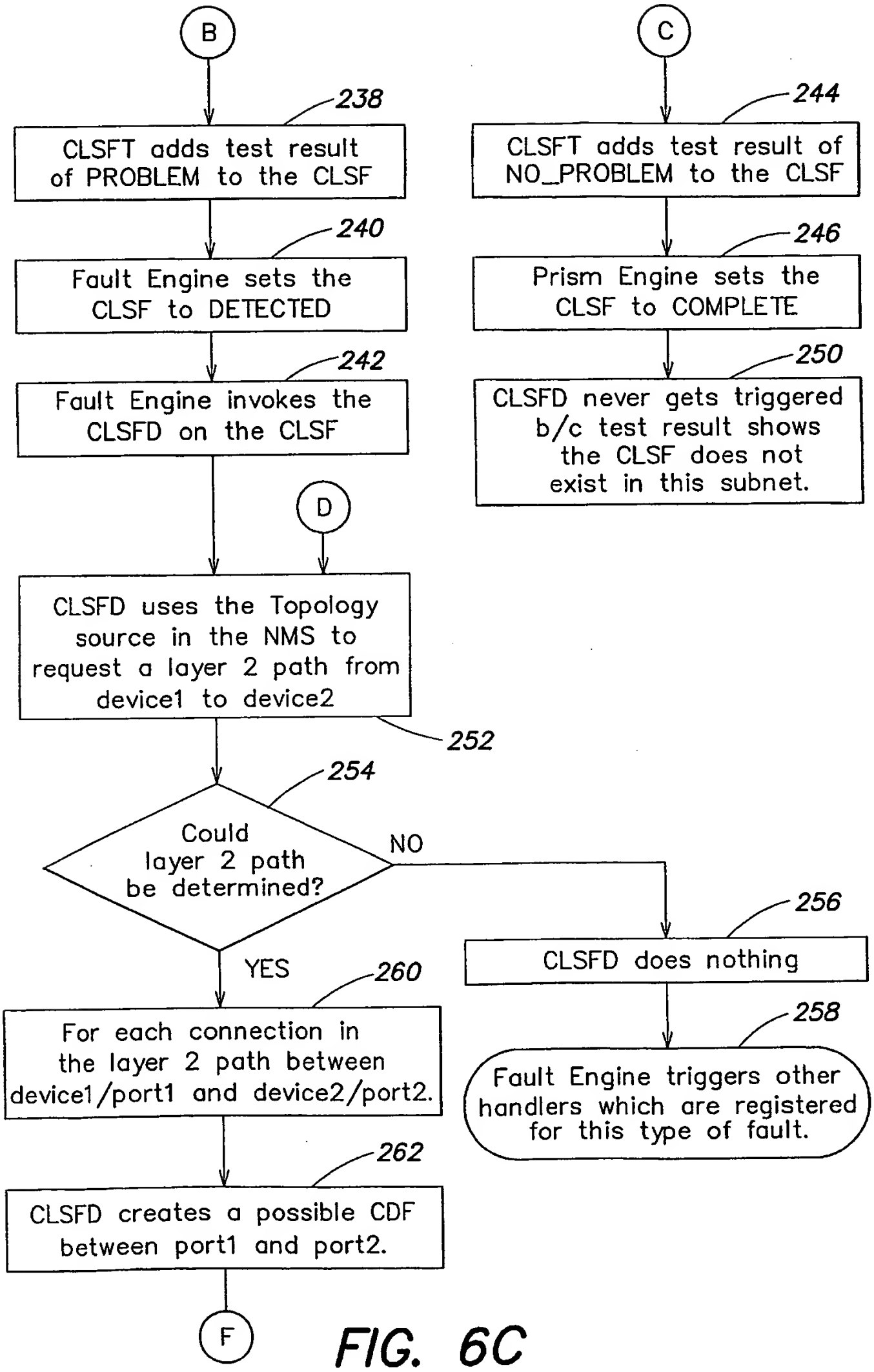

B
238
CLSFT adds test result of PROBLEM to the CLSF
240
Fault Engine sets the CLSF to DETECTED
242
Fault Engine invokes the CLSFD on the CLSF
D
CLSFD uses the Topology source in the NMS to request a layer 2 path from device1 to device2
252
254
Could layer 2 path be determined?
NO
YES
256
CLSFD does nothing
258
Fault Engine triggers other handlers which are registered for this type of fault.
260
For each connection in the layer 2 path between device1/port1 and device2/port2.
262
CLSFD creates a possible CDF between port1 and port2.
F
C
244
CLSFT adds test result of NO_PROBLEM to the CLSF
246
Prism Engine sets the CLSF to COMPLETE
250
CLSFD never gets triggered b/c test result shows the CLSF does not exist in this subnet.

FIG. 6C

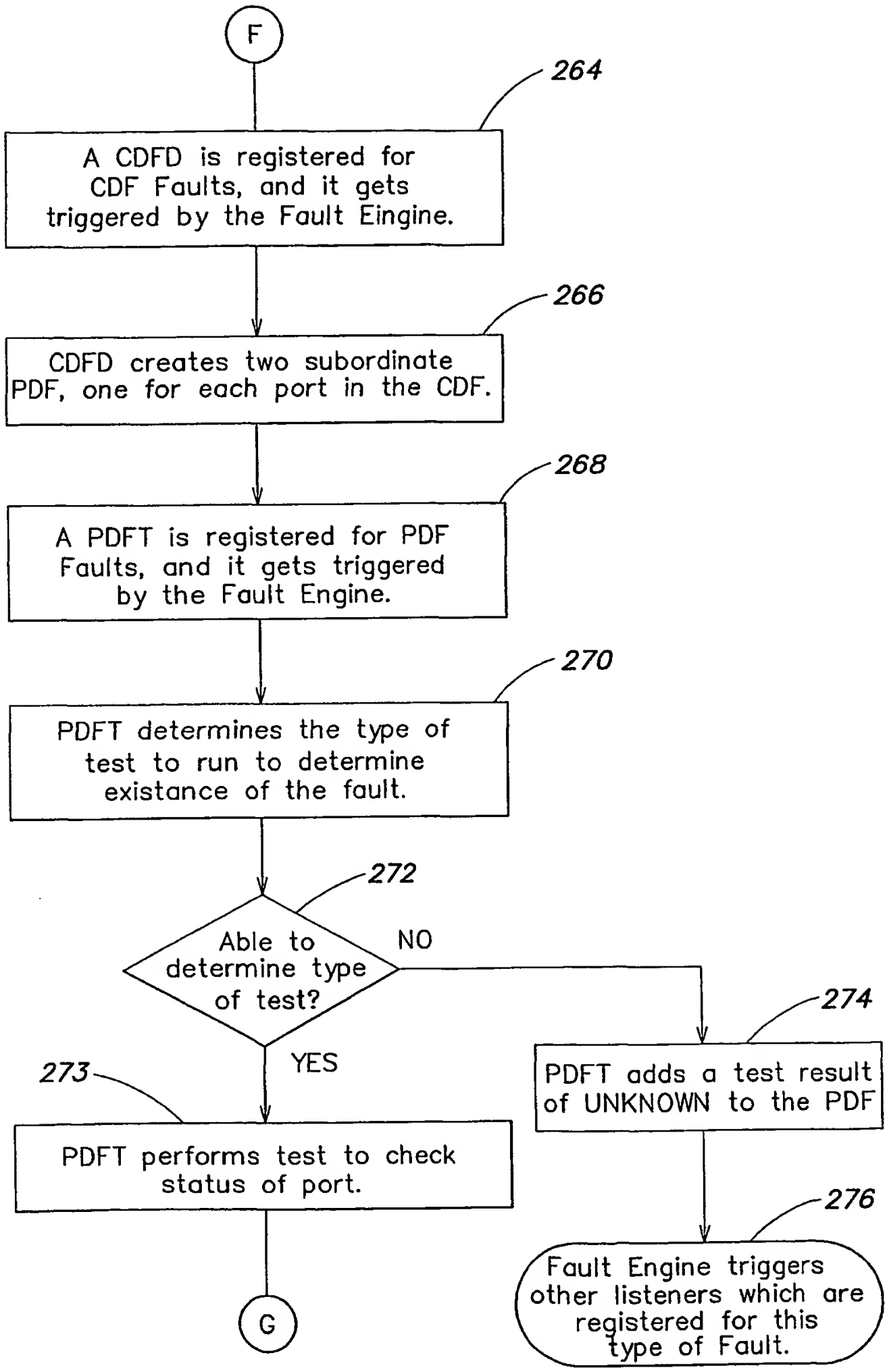

F
264
A CDFD is registered for CDF Faults, and it gets triggered by the Fault Eingine.
266
CDFD creates two subordinate PDF, one for each port in the CDF.
268
A PDFT is registered for PDF Faults, and it gets triggered by the Fault Engine.
270
PDFT determines the type of test to run to determine existance of the fault.
272
Able to determine type of test?
NO
YES
273
PDFT performs test to check status of port.
G
274
PDFT adds a test result of UNKNOWN to the PDF
276
Fault Engine triggers other listeners which are registered for this type of Fault.

FIG. 6D

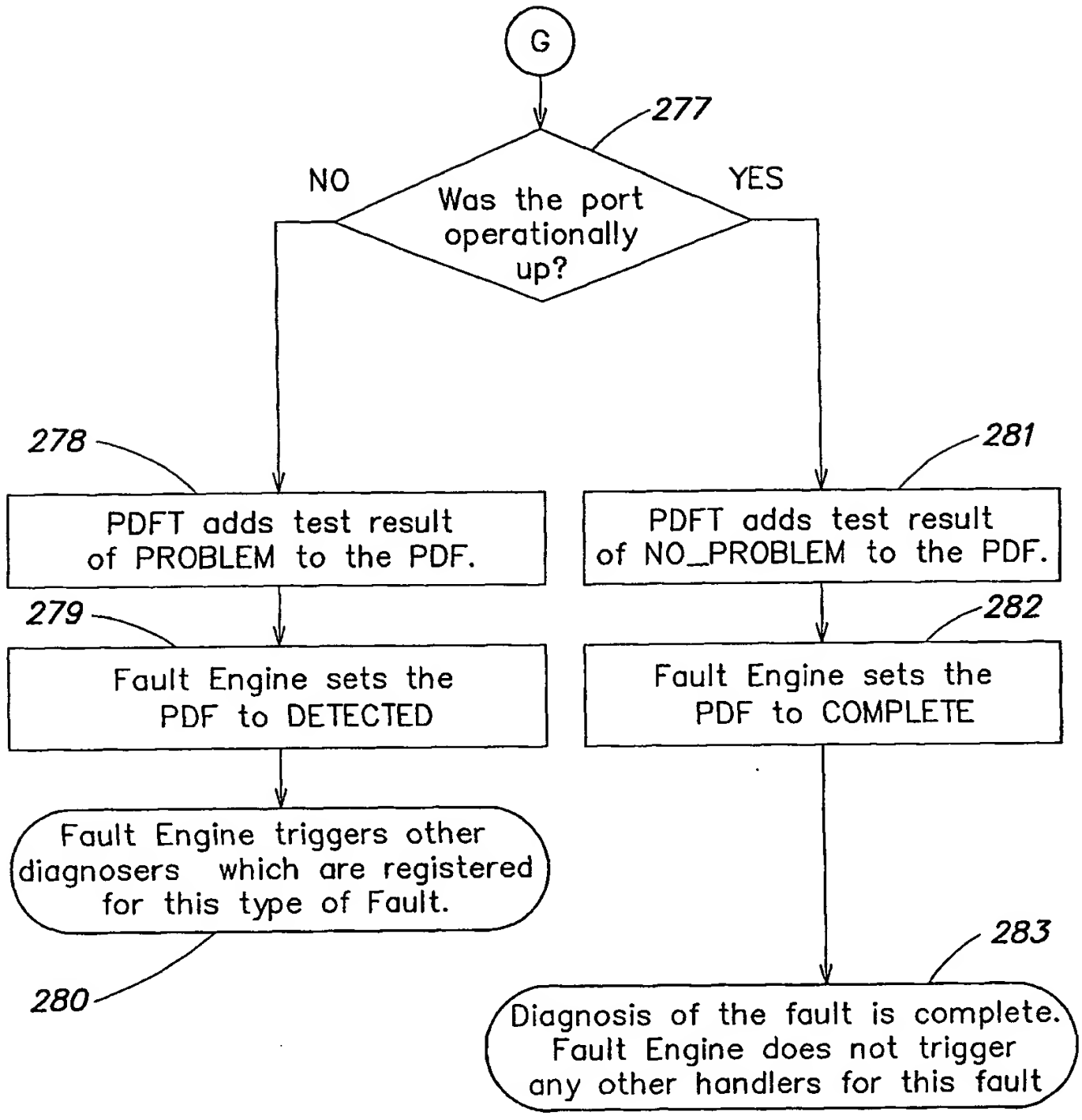

G
277
NO
YES
Was the port operationally up?
278
PDFT adds test result of PROBLEM to the PDF.
279
Fault Engine sets the PDF to DETECTED
Fault Engine triggers other diagnosers which are registered for this type of Fault.
280
281
PDFT adds test result of NO_PROBLEM to the PDF.
282
Fault Engine sets the PDF to COMPLETE
283
Diagnosis of the fault is complete. Fault Engine does not trigger any other handlers for this fault

FIG. 6E

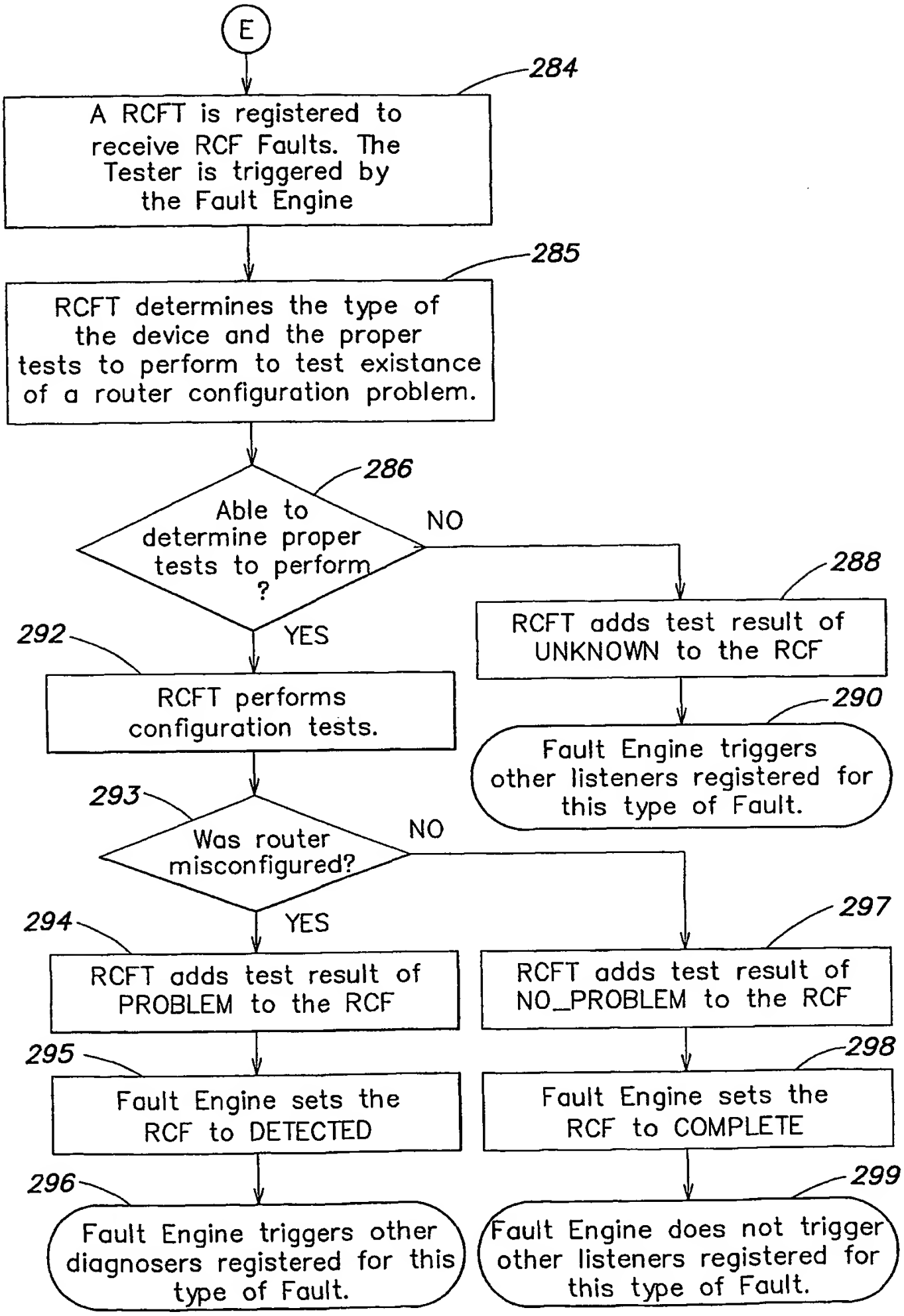

E
284
A RCFT is registered to receive RCF Faults. The Tester is triggered by the Fault Engine
285
RCFT determines the type of the device and the proper tests to perform to test existance of a router configuration problem.
286
Able to determine proper tests to perform ?
NO
YES
288
RCFT adds test result of UNKNOWN to the RCF
290
Fault Engine triggers other listeners registered for this type of Fault.
292
RCFT performs configuration tests.
293
Was router misconfigured?
NO
YES
294
RCFT adds test result of PROBLEM to the RCF
295
Fault Engine sets the RCF to DETECTED
296
Fault Engine triggers other diagnosers registered for this type of Fault.
297
RCFT adds test result of NO_PROBLEM to the RCF
298
Fault Engine sets the RCF to COMPLETE
299
Fault Engine does not trigger other listeners registered for this type of Fault.

FIG. 6F

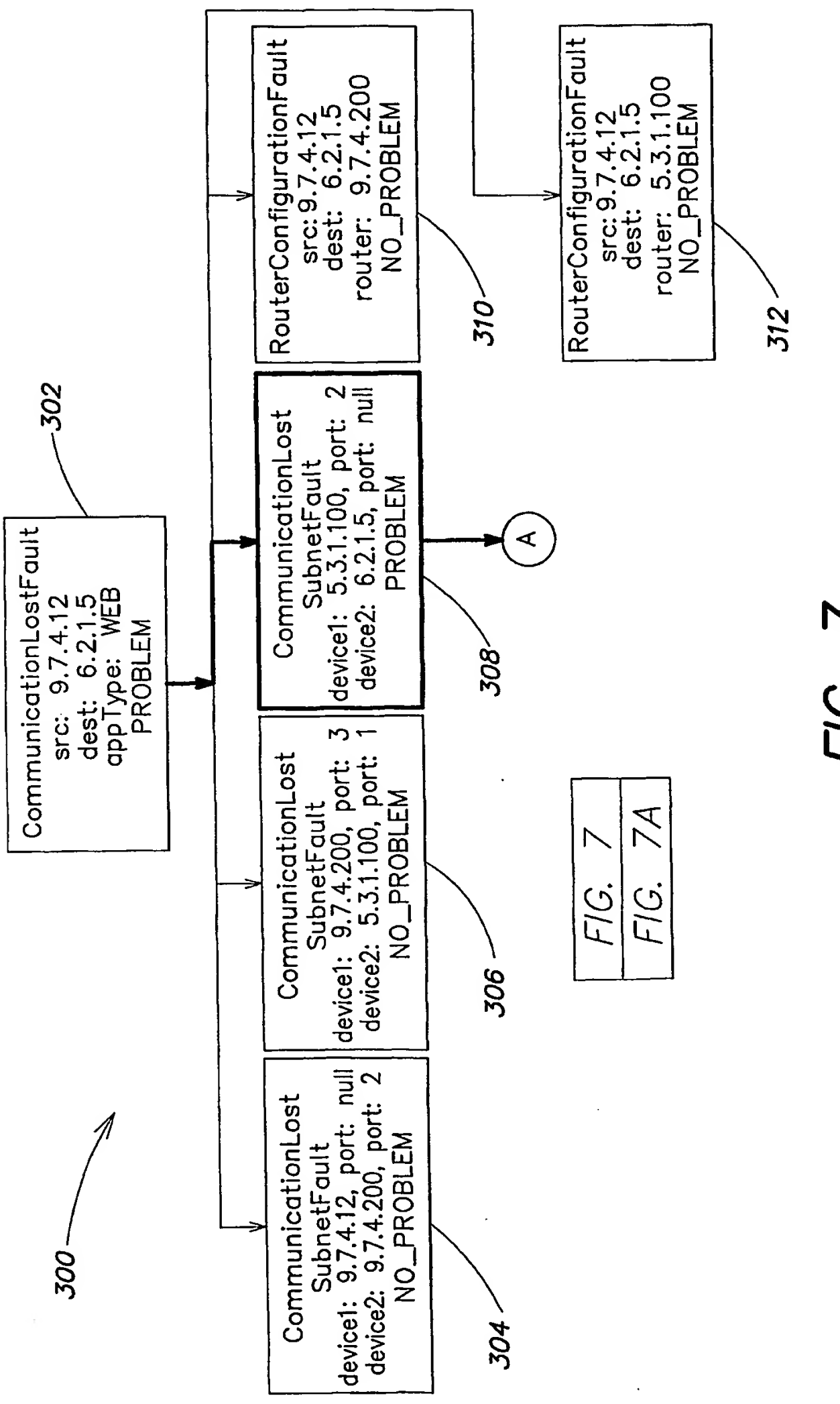


FIG. 7

FIG. 7
FIG. 7A

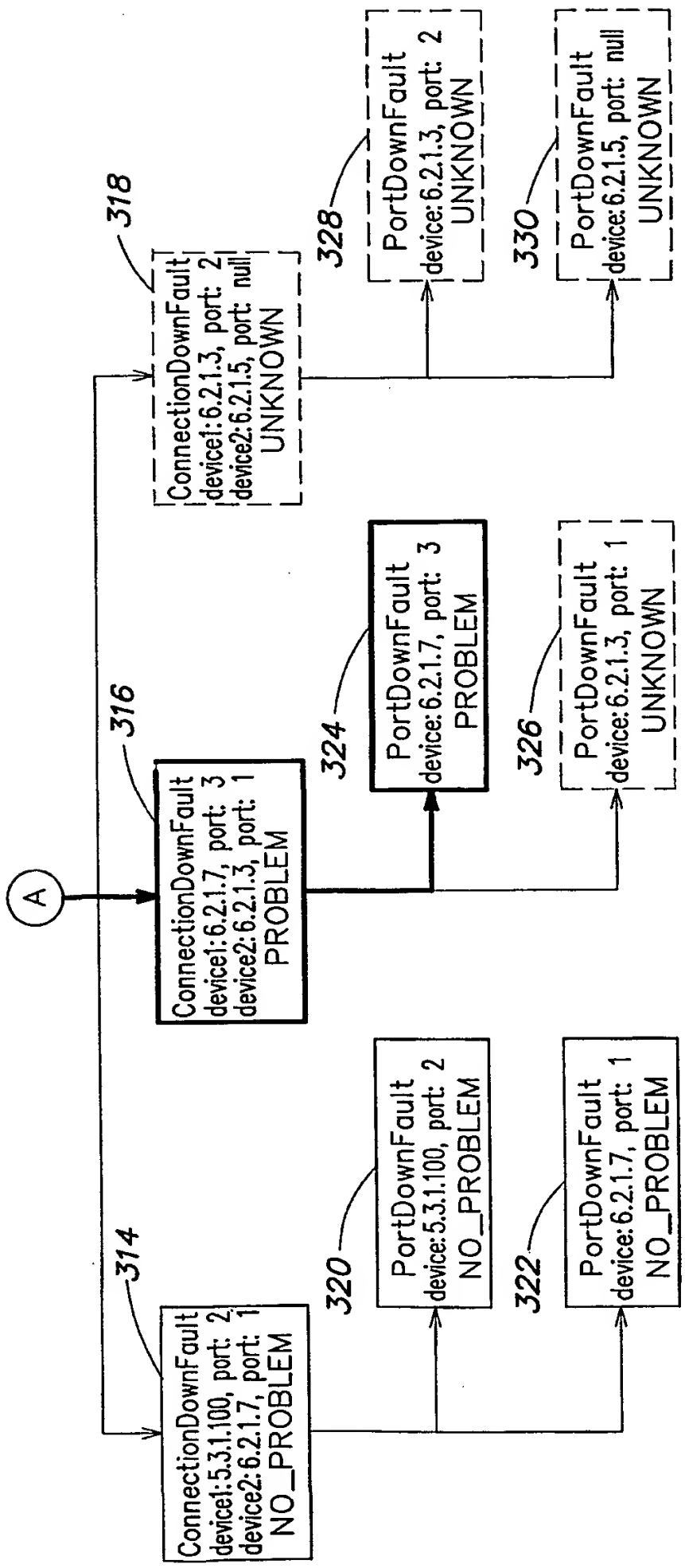

A
316
ConnectionDownFault
device1: 5.3.1.100, port: 2
device2: 6.2.1.7, port: 1
NO_PROBLEM
318
ConnectionDownFault
device1: 6.2.1.7, port: 3
device2: 6.2.1.3, port: 1
PROBLEM
314
ConnectionDownFault
device1: 5.3.1.100, port: 2
device2: 6.2.1.7, port: 1
NO_PROBLEM
320
PortDownFault
device: 5.3.1.100, port: 2
NO_PROBLEM
322
PortDownFault
device: 6.2.1.7, port: 1
NO_PROBLEM
324
PortDownFault
device: 6.2.1.7, port: 3
PROBLEM
326
PortDownFault
device: 6.2.1.3, port: 1
UNKNOWN
328
PortDownFault
device: 6.2.1.3, port: 2
UNKNOWN
330
PortDownFault
device: 6.2.1.5, port: null
UNKNOWN
300

FIG. 7A

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/14618

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : G06F 11/00
US CL : 714/43, 46
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : 714/43,46

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
US Patfull

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5,864,662 A (BROWNMILLER et al.) 26 January 1999 (25.01.99) Column 3	1-41

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search
02 July 2001 (02.07.2001)

Date of mailing of the international search report
14 AUG 2001

Name and mailing address of the ISA/US
Commissioner of Patents and Trademarks
Box PCT
Washington, D.C. 20231
Facsimile No. (703)305-3230

Authorized officer
Robert Beausoliel
Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet) (July 1998)